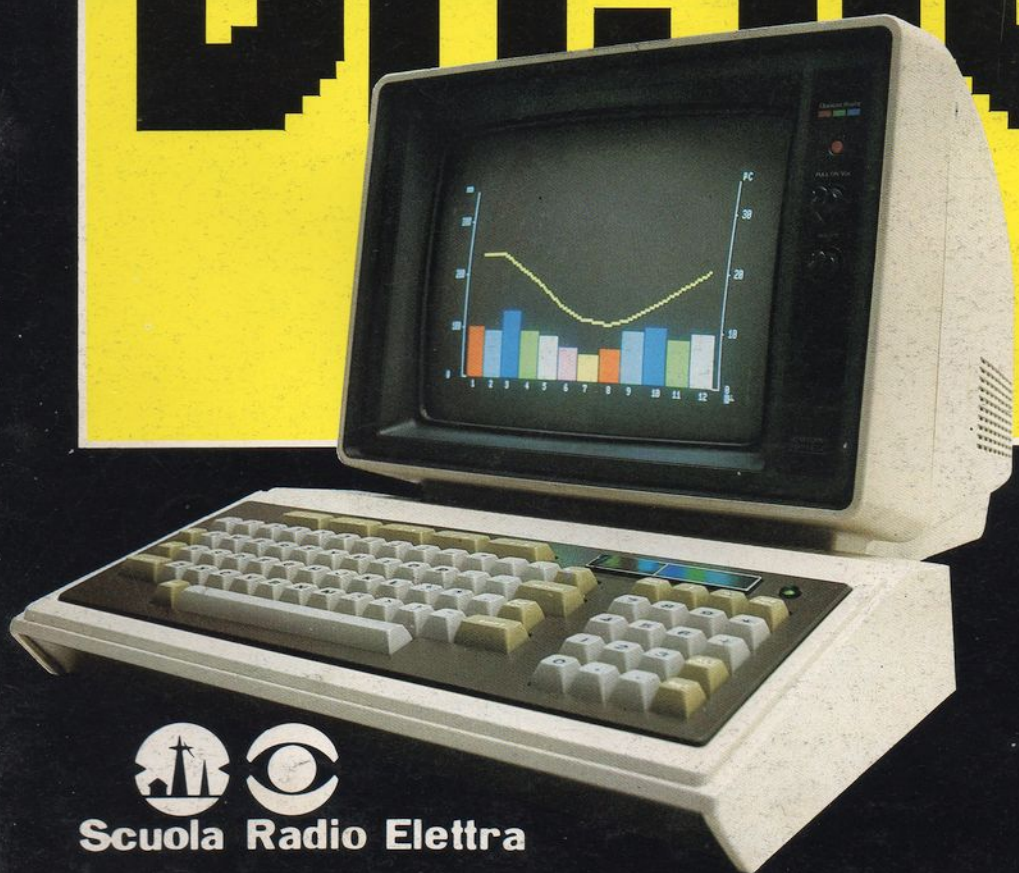
PARLA BASIC



Scuola Radio Elettra

Scuola Radio Elettra
Torino

Analisi e programmazione

Lezione 10

1 — La subroutine
2 — Le funzioni di libreria e l'istruzione **DEF FN**
3 — Istruzioni per operare direttamente su matrici e vettori

Che cosa apprenderà

In questa lezione saranno ripresi ed ampliati, mediante appositi esempi di programmazione, i concetti di subroutine (sottoprogramma).

Verrà poi completata la trattazione delle funzioni matematiche già programmate nel computer (denominate anche funzioni di libreria), che l'operatore può all'occorrenza richiedere con un apposito nome.

Si verificherà, inoltre, l'operatività dell'istruzione **DEF FN** il cui scopo è di consentire il calcolo di qualsiasi altra funzione matematica definita dal programmatore.

Infine, verranno prese in considerazione le istruzioni presenti in alcuni linguaggi BASIC piú evoluti, che permettono di operare direttamente sulle matrici e sui vettori.

1. – La subroutine

Come già accennato in una precedente lezione, la subroutine o sottoprogramma è costituita da un gruppo di istruzioni che nel corso dell'elaborazione viene ripetuto piú volte.

Da un punto di vista formale la subroutine è perfettamente simile ad un programma.

Essa è cioè costituita da un gruppo di istruzioni numerate, caratterizzate alla fine dalla presenza di un enunciato BASIC particolare: **RETURN**, che serve per il ritorno automatico al programma principale.

Quando dal programma principale si vuole saltare al sottoprogramma, si scrive l'istruzione BASIC **GOSUB**, seguita dal numero della prima linea del sottoprogramma. Quest'ultimo ritornerà automaticamente, attraverso l'istruzione **RETURN**, nel programma principale all'istruzione successiva alla **GOSUB**.

Si può raffigurare schematicamente una subroutine come indicato nella *fig. 1*, dove appaiono anche le istruzioni **GOSUB** e **RETURN** che servono a collegare la subroutine stessa a diversi punti del programma principale.

Come si vede dalla *fig. 1*, la subroutine di nome "SUB A" inizia al numero di linea indicato genericamente con N1 e viene "chiamata" (o elaborata) due volte dal programma principale da due punti diversi; il ritorno (istruzione **RETURN** della subroutine) avviene nei due punti diversi indicati con R1 e R2.

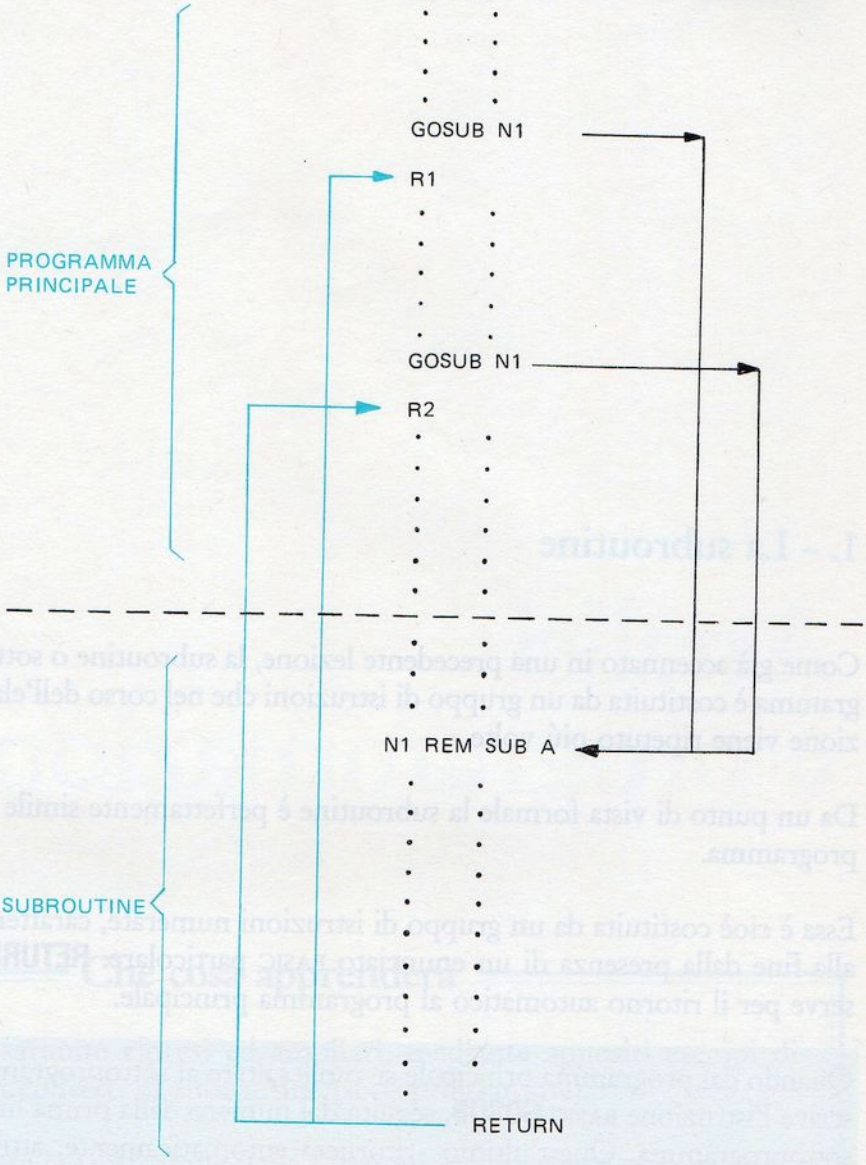


Fig. 1 – Rappresentazione schematica di un programma contenente una subroutine di nome "SUB A", iniziante al numero N1.

Non è necessario dare un nome alla subroutine, la quale normalmente risiede nella parte finale del programma principale. I numeri di riferimento delle subroutine sono diversi da quelli del programma principale.

Possono essere scritte piú di una subroutine in uno stesso programma principale ed ogni subroutine può saltare ad altre subroutine, secondo uno schema "nidificato", analogo a quello visto per i cicli di programma utilizzanti le istruzioni **FOR... NEXT....**

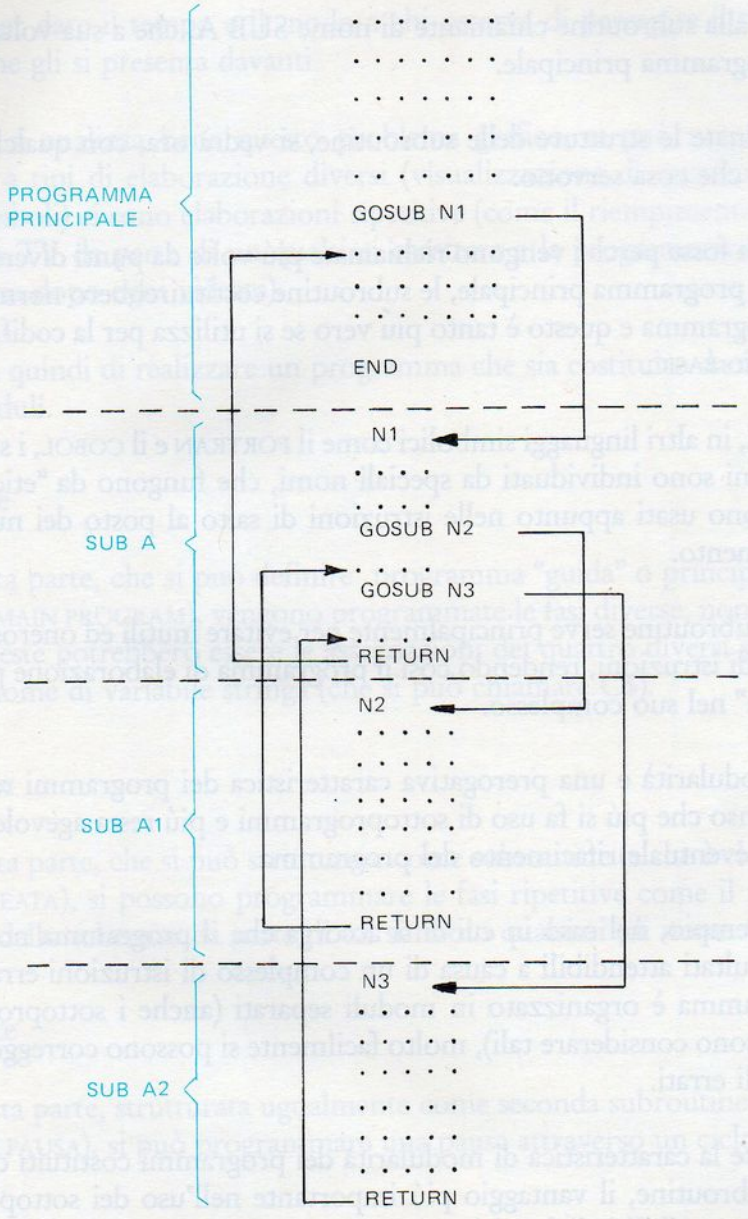


Fig. 2 – Schema di un programma utilizzante una subroutine SUB A ed "indirettamente" altre due subroutine "nidificate" SUB A1 e SUB A2, che iniziano rispettivamente ai numeri di linea N2 e N3.

Nello schema della *fig. 2* è riportata una possibile configurazione di programma utilizzante una subroutine (di nome SUB A), che a sua volta utilizza altre due subroutine (di nome SUB A1 e SUB A2).

Come si vede, queste ultime non ritornano al programma principale,

bensí alla subroutine chiamante di nome SUB A, che a sua volta ritorna al programma principale.

Esaminate le strutture delle subroutine, si vedrà ora, con qualche esempio, a che cosa servono.

Se non fosse perché vengono richiamate piú volte da punti diversi di uno stesso programma principale, le subroutine costituirebbero normali parti di programma e questo è tanto piú vero se si utilizza per la codifica il linguaggio BASIC.

Infatti, in altri linguaggi simbolici come il FORTRAN e il COBOL, i sottoprogrammi sono individuati da speciali nomi, che fungono da "etichetta" e che sono usati appunto nelle istruzioni di salto al posto dei numeri di riferimento.

Una subroutine serve principalmente per evitare inutili ed onerose ripetizioni di istruzioni, rendendo cosí il programma di elaborazione piú "modulare" nel suo complesso.

La modularità del programma

La modularità è una prerogativa caratteristica dei programmi razionali, nel senso che piú si fa uso di sottoprogrammi e piú resta agevole l'opera di un eventuale rifacimento del programma.

Per esempio, nel caso in cui ci si accorga che il programma non fornisce risultati attendibili a causa di un complesso di istruzioni errate, se il programma è organizzato in moduli separati (anche i sottoprogrammi si possono considerare tali), molto facilmente si possono correggere i soli moduli errati.

A parte la caratteristica di modularità dei programmi costituiti da una o piú subroutine, il vantaggio piú importante nell'uso dei sottoprogrammi è la possibilità di far elaborare piú volte uno stesso calcolo dallo stesso gruppo di istruzioni (quelle appunto che compongono la subroutine): elaborazioni ripetitive cioè che ricorrono nello stesso sottoprogramma anziché servirsi di inutili e prolissi rifacimenti di istruzioni.

Per fare un esempio grafico, si supponga di voler riempire lo schermo del televisore con caratteri grafici come quattro diverse lettere dell'alfabeto o, per chi possiede un microcomputer Commodore, i quattro simboli (semi) delle carte da gioco: ♠ (picche) ♥ (cuori) ♦ (quadri) e ♣ (fiori).

Tra una videata e l'altra inoltre si supponga di volere interporre una certa

pausa, per dare il tempo e il modo a chi osserva di percepire il tipo di figura che gli si presenta davanti.

Ora, se si analizza bene questo problema grafico, si può notare che accanto a tipi di elaborazione diversi (visualizzazione di quattro differenti simboli) ci sono elaborazioni ripetitive (come il riempimento dello schermo TV da parte di un qualsiasi carattere e la programmazione di una pausa dopo ogni videata).

Si tratta quindi di realizzare un programma che sia costituito da tre parti o moduli.

1ª Parte

In questa parte, che si può definire programma "guida" o principale (in inglese MAIN PROGRAM), vengono programmate le fasi diverse, non ripetitive: queste potrebbero essere le assegnazioni dei quattro diversi simboli ad un nome di variabile stringa (che si può chiamare C$).

2ª Parte

In questa parte, che si può strutturare come subroutine (e che è chiamata SUB VIDEATA), si possono programmare le fasi ripetitive come il riempimento dello schermo da parte di un simbolo qualsiasi (di nome C$).

3ª Parte

In questa parte, strutturata ugualmente come seconda subroutine (di nome SUB PAUSA), si può programmare una pausa attraverso un ciclo **FOR... NEXT....**

La prima parte guida e controlla le altre due parti del programma.

Nella *fig. 3* è rappresentato il flow-chart di questo problema grafico e nella *fig. 4* la relativa codifica in linguaggio BASIC (si è supposto di utilizzare il microsistema C64, dove appunto sono contemplati i simboli grafici ♠, ♥, ♦, ♣).

Come si può rilevare dalla *fig. 4*, il listato del programma è molto semplice.

Nelle istruzioni 3Ø, 45, 6Ø e 75 sono assegnate alla stessa variabile

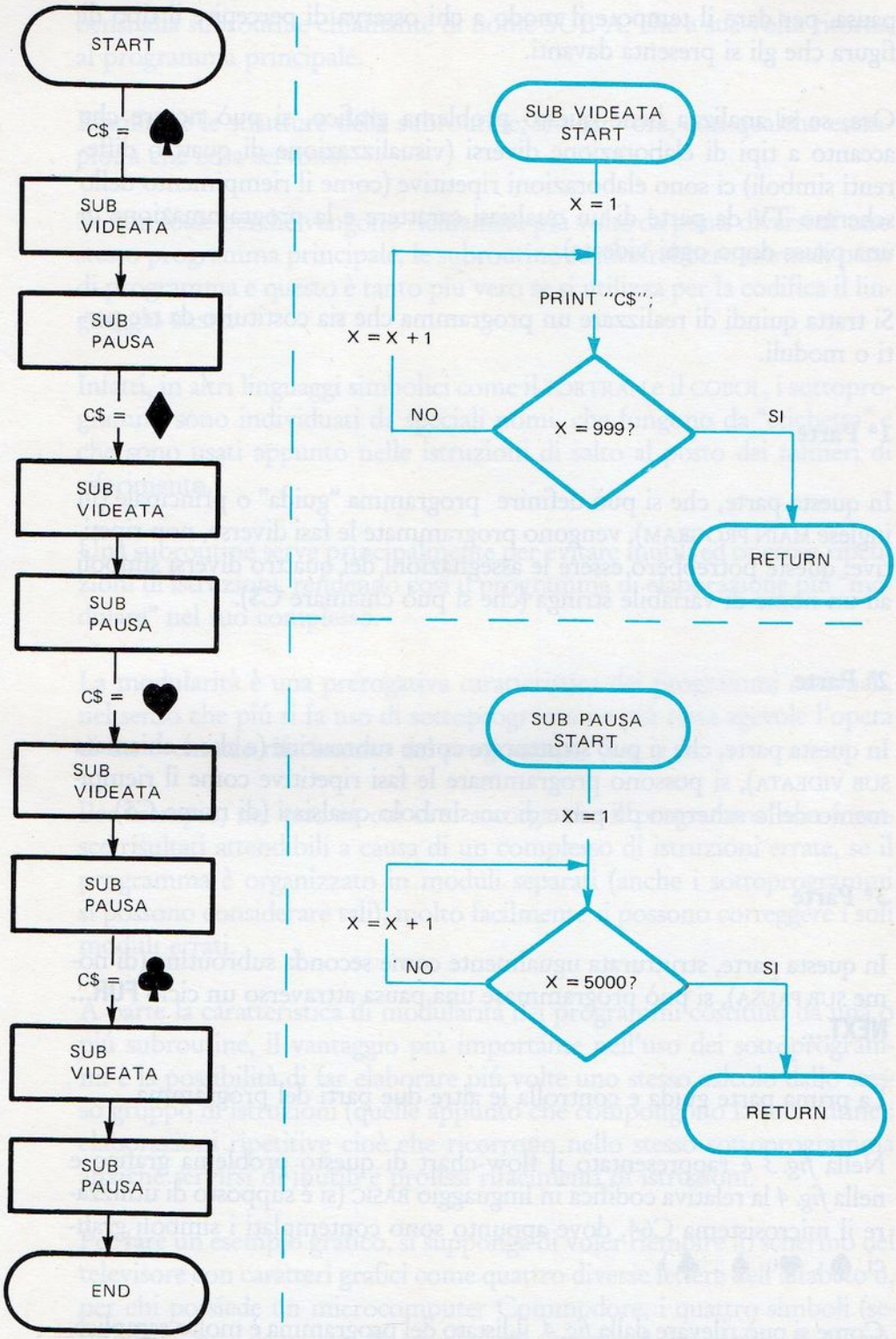


Fig. 3 – Flow-chart del programma grafico che visualizza, ad intervalli regolari di tempo, i semi delle carte da gioco. Le subroutine sono diagrammate in moduli separati.

```
10 REM***************
12 REM*SIMBOLI CARTE*
15 REM*   DA GIOCO   *
20 REM***************
25 PRINT"■"
30 LET C$="♠"
35 GOSUB 100
40 GOSUB 120
45 LET C$="♣"
50 GOSUB 100
55 GOSUB 120
60 LET C$="♥"
65 GOSUB 100
70 GOSUB 120
75 LET C$="♦"
80 GOSUB 100
85 GOSUB 120
90 END
100 REM**SUB VIDEATA**
105 FOR X=1 TO 999
110 PRINT C$;
115 NEXT:RETURN
120 REM**SUB PAUSA**
125 FOR X=1 TO 5000:NEXT
127 PRINT"■"
130 RETURN
```

Fig. 4 – Codifica in linguaggio BASIC *(per C64) del programma che visualizza sullo schermo i simboli delle carte da gioco.*

stringa di nome C$ i quattro diversi simboli delle carte da gioco da visualizzare sullo schermo (oppure quattro lettere diverse dell'alfabeto).

Dopo ogni assegnazione sono previste le chiamate (**GOSUB**) di due subroutine, che iniziano rispettivamente alle linee di programma 100 e 120.

Nella prima subroutine (istruzione 100) è presente un ciclo **FOR... NEXT...**, che provvede a fare stampare 999 volte sullo schermo ciascuno dei quattro simboli delle carte da gioco (Computer C64).

Dopo ciò l'elaborazione ritorna al programma principale mediante l'istruzione **RETURN**, presente alla linea di programma 115.

Nei microcomputer Commodore i quattro simboli delle carte da gioco sono ottenuti premendo contemporaneamente il tasto **SHIFT** ed il tasto che riporta sul lato anteriore il simbolo della carta interessata.

Il ciclo **FOR... NEXT...** in questione conta da 1 a 999 anziché da 1 a 1000 quante sono le posizioni di stampa sullo schermo del C64, poiché se si visualizzasse il millesimo carattere grafico quelli già presenti sullo schermo scorrerebbero verso l'alto di una riga e quindi verrebbero riprodotti meno simboli.

Utilizzando per questo programma il microcomputer VIC-20, occorre modificare l'istruzione 105 come segue:

```
105 FOR X=1 TO 505
```

Anche in questo caso l'iterazione conta da 1 a 505 (anziché da 1 a 506, quante sono le posizioni di stampa sullo schermo del VIC-20) per non avere lo scorrimento verso l'alto di quanto visualizzato sullo schermo stesso.

La seconda subroutine (istruzioni 120÷130) crea un'iterazione di ritardo, che consente di ottenere, come già precisato, una pausa fra una videata e l'altra di simboli grafici.

Per avere pause piú lunghe o piú corte basta aumentare o diminuire l'estremo superiore della **FOR** riportata alla linea di programma 125.

Per eseguire il programma in questione, dopo averlo immesso nel computer, basta battere il comando **RUN**: si dovrebbero vedere quattro videate, ognuna delle quali costituita da uno dei simboli delle carte da gioco.

Se si fosse scritto il programma precedente senza far uso delle due subroutine, si sarebbero dovute programmare ben quattro volte le videate dei simboli grafici (dopo le relative istruzioni di assegnazione **LET** C$...) e le pause temporali (dopo ogni videata).

Dall'esposizione di questo semplice problema si può quindi constatare come siano importanti le subroutine in un programma che contenga elaborazioni di tipo "iterativo" (termine che nel gergo dell'informatica significa "ripetitivo").

Di grande utilità sono inoltre le subroutine soprattutto nei calcoli matematici e nelle elaborazioni di tipo gestionale-amministrativo.

Per calcolare ad esempio le fatture di un grande numero di clienti, si progetta un programma di elaborazione composto essenzialmente di due parti: una prima parte principale che richiama, da qualche archivio o file

registrato su un'unità di memoria ausiliaria, le informazioni generali relative ad ogni cliente (come CODICE, NOMINATIVO, INDIRIZZO, CODICE FISCALE, PARTITA I.V.A.), e una seconda parte che, combinando queste ultime con i movimenti (acquisti) effettuati nell'arco di un certo periodo di tempo, provvede successivamente a richiamare una "subroutine" di fatturazione.

Questa "subroutine", richiamata di volta in volta per ogni cliente, valorizza le quantità con i prezzi degli articoli acquistati, calcolando anche l'I.V.A. secondo varie percentuali.

La stessa subroutine provvede a stampare su appositi moduli, predisposti in rotoli nella stampante periferica, i prospetti di fatturazione che devono essere riportati secondo le indicazioni fornite.

1.1 - Caratteristiche di una subroutine

Una subroutine, come si è visto, può considerarsi parte integrante di un programma, anche se di solito viene codificata in fondo a quest'ultimo perché venga meglio evidenziata.

Le elaborazioni ripetitive effettuate dalla subroutine utilizzano i valori numerici o stringa già predisposti (o assegnati) nel programma principale in certi nomi di variabili.

I risultati delle elaborazioni ripetitive, a loro volta, vengono "depositati" dalla subroutine in altri nomi di variabili, che poi verranno utilizzati dal programma principale.

In alcuni linguaggi di programmazione, come il FORTRAN ed il COBOL, i nomi di variabili usati in "comune" da un programma principale e da una subroutine vanno specificati con un'istruzione particolare chiamata **COMMON**.

In una subroutine di linguaggio BASIC non si possono ridimensionare vettori e matrici già dimensionati nel programma principale.

Un'altra caratteristica importante di una subroutine è quella di poter essere utilizzata a seguito di una diramazione condizionata che, come si è visto nelle precedenti lezioni, si effettua scrivendo un'istruzione BASIC **IF... THEN...** (**GOTO**) oppure **ON... GOTO...**.

Nel nostro caso, in entrambe le istruzioni condizionali suddette la parola chiave **GOTO** verrebbe sostituita con la parola chiave di salto **GOSUB**, che appunto consente di deviare l'elaborazione alla prima istruzione di una determinata subroutine (l'istruzione cioè che ha il numero di linea uguale a quello riportato subito dopo la parola chiave **GOSUB**).

Dopo l'elaborazione della subroutine, il ritorno naturalmente sarà nel punto successivo alla **IF... THEN... GOSUB**, oppure alla **ON... GOSUB**.

Un esempio di utilizzazione dell'istruzione **ON... GOSUB** è stato visto nella lezione *Pratica del* BASIC *6*, dove è stato realizzato un programma BASIC che calcolava l'area ed il perimetro dei primi quattro poligoni regolari (triangolo equilatero, quadrato, pentagono ed esagono regolari) dopo aver introdotto da tastiera la lunghezza del lato L comune ed il numero N dei lati del poligono.

Come ulteriore esempio d'impiego delle subroutine si può provare a scrivere un altro programma BASIC, che ha scopi ed usi commerciali e può essere utilizzato da persone che esercitano qualche attività di vendita al minuto.

Si supponga che un negoziante di articoli di abbigliamento venda al pubblico praticando due diversi sconti commerciali: lo sconto del 10% se il valore di quanto si è acquistato è compreso fra due ammontari, per esempio fra 100.000 lire e 200.000 lire, e lo sconto del 12% se il valore dell'acquisto supera 200.000 lire. Non si pratica alcuno sconto invece se il valore è inferiore a 100.000 lire.

Si tratta di scrivere un programma BASIC che preveda l'immissione da tastiera dell'importo pari al valore dell'acquisto fatto dal cliente (valore che va calcolato a parte con altri supporti, per esempio con una calcolatrice tascabile) e che successivamente pratichi sul totale lo sconto adeguato all'entità dell'importo. Nella *fig. 5* e nella *fig. 6* sono riportati rispettivamente il flow-chart e la codifica del programma in questione.

Nell'istruzione 3Ø il programma si chiede se l'importo V (pari al valore della merce acquistata dal cliente) è compreso fra 1ØØ.ØØØ lire e 2ØØ.ØØØ lire. Se tale importo è inferiore a 1ØØ.ØØØ lire, sullo schermo compare la scritta

"L'IMPORTO DA PAGARE E'"

seguita dalla cifra pari al valore della merce acquistata senza alcuno sconto.

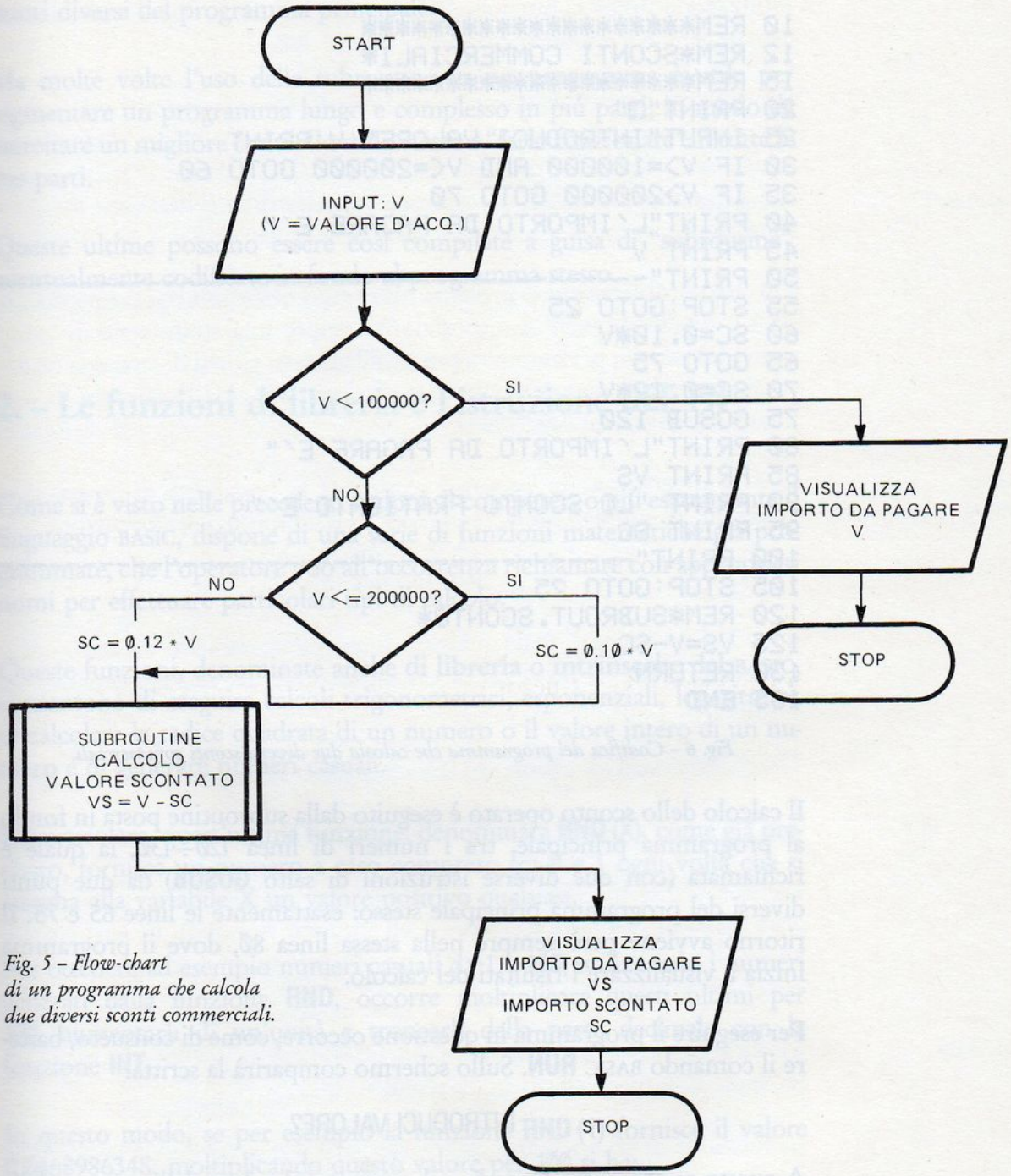


Fig. 5 – Flow-chart di un programma che calcola due diversi sconti commerciali.

Se invece l'importo è compreso fra i due valori indicati, il programma salta all'istruzione 6Ø, nella quale viene assegnato alla variabile SC il valore Ø.1Ø*V (sconto del 10%). Nel caso invece il valore dell'importo sia superiore a 2ØØ.ØØØ lire, l'elaborazione salta alla linea di programma 7Ø, dove alla variabile SC viene assegnato il valore Ø.12*V (sconto del 12%).

```
10 REM**********************
12 REM*SCONTI COMMERCIALI*
15 REM**********************
20 PRINT"□"
25 INPUT"INTRODUCI VALORE";V:PRINT
30 IF V>=100000 AND V<=200000 GOTO 60
35 IF V>200000 GOTO 70
40 PRINT"L'IMPORTO DA PAGARE E'"
45 PRINT V
50 PRINT"-------------------------------------------------"
55 STOP:GOTO 25
60 SC=0.10*V
65 GOTO 75
70 SC=0.12*V
75 GOSUB 120
80 PRINT"L'IMPORTO DA PAGARE E'"
85 PRINT VS
90 PRINT "LO SCONTO PRATICATO E'"
95 PRINT SC
100 PRINT"-------------------------------------------------"
105 STOP:GOTO 25
120 REM*SUBROUT.SCONTO*
125 VS=V-SC
130 RETURN
135 END
```

Fig. 6 – Codifica del programma che calcola due diversi sconti commerciali.

Il calcolo dello sconto operato è eseguito dalla subroutine posta in fondo al programma principale, tra i numeri di linea 12Ø÷13Ø, la quale è richiamata (con due diverse istruzioni di salto **GOSUB**) da due punti diversi del programma principale stesso: esattamente le linee 65 e 75. Il ritorno avviene però sempre nella stessa linea 8Ø, dove il programma inizia a visualizzare i risultati del calcolo.

Per eseguire il programma in questione occorre, come di consueto, battere il comando BASIC **RUN**. Sullo schermo comparirà la scritta:

INTRODUCI VALORE?

A questo punto, introducendo nel computer il valore dell'importo della merce acquistata, sullo schermo compariranno le scritte che specificano l'importo da pagare e lo sconto praticato (nel caso che il valore dell'acquisto superi le 1ØØ.ØØØ lire).

Si è visto che il maggiore utilizzo delle subroutine in un programma si verifica quando occorre ripetere uno stesso ciclo di operazioni da piú

punti diversi del programma principale.

Ma molte volte l'uso delle subroutine in un programma è utile per segmentare un programma lungo e complesso in piú parti, in modo da esercitare un migliore controllo sullo stesso e "farlo funzionare" in tutte le sue parti.

Queste ultime possono essere cosí compilate a guisa di "subroutine", eventualmente codificate in fondo al programma stesso.

2. – Le funzioni di libreria e l'istruzione DEF FN

Come si è visto nelle precedenti lezioni, il computer, o piú esattamente il linguaggio BASIC, dispone di una serie di funzioni matematiche già programmate, che l'operatore può all'occorrenza richiamare con appropriati nomi per effettuare particolari tipi di calcolo.

Queste funzioni, denominate anche **di libreria** o **intrinseche del** BASIC, consentono di eseguire calcoli trigonometrici, esponenziali, logaritmici, di calcolare la radice quadrata di un numero o il valore intero di un numero e di generare numeri casuali.

In particolare, quest'ultima funzione, denominata **RND** (X), come già precisato, fornisce un numero a caso compreso fra Ø e 1 ogni volta che si assegna alla variabile X un valore positivo qualsiasi.

Per ottenere ad esempio numeri casuali da 1 a 1ØØ, utilizzando i numeri generati dalla funzione **RND**, occorre moltiplicare questi ultimi per 1ØØ, aumentarli di un'unità e troncarli della parte decimale con la funzione **INT**.

In questo modo, se per esempio la funzione **RND** (1) fornisce il valore Ø.Ø468986348, moltiplicando questo valore per 1ØØ si ha:

4.68986348

e sommando ad esso il numero 1 si ottiene:

5.68986348

Utilizzando infine la funzione **INT** (5.68986348), si ottiene il valore intero casuale 5, che è appunto compreso fra 1 e 1ØØ.

Il problema di trasformare i numeri casuali (compresi tra Ø e 1) forniti dal computer in altri numeri compresi tra altri due valori viene quindi risolto semplicemente moltiplicando il valore casuale per il valore massimo di questi due valori ed aggiungendo al risultato ottenuto il valore minimo.

Un altro problema che si presenta nel campo della simulazione probabilistica è quello di avere a disposizione sempre una stessa serie di valori numerici casuali; per esempio, per simulare certi giochi di strategia occorre disporre di serie numeriche casuali ripetitive, cioè verificatesi in momenti e circostanze precedenti.

L'istruzione Randomize

Ora in diversi sistemi personal computer la possibilità di creare serie "ripetute" di numeri casuali è consentita dall'utilizzo dell'istruzione

RANDOMIZE (X)

dove X è un numero reale qualsiasi, il cui valore serve in un certo senso ad "inizializzare" il processo di casualizzazione dei numeri generati dal computer.

Diversi altri sistemi utilizzano invece numeri X negativi nella funzione:

RND (X)

per ricavare serie ripetute di numeri casuali (scrivendo successivamente altre funzioni **RND** (X), in cui X è un numero positivo). In altri sistemi ancora il valore di X che consente di riottenere un numero casuale precedentemente estratto è Ø.

Funzione ABS

Molte volte nei calcoli si desidera utilizzare il valore assoluto di un numero, ossia il numero stesso senza segno + o segno –.

La funzione di libreria BASIC che consente di realizzare questa operazione e la funzione **ABS** (X), la quale fornisce appunto il valore assoluto di X.

Per esempio, scrivendo:

PRINT ABS (–45)

si otterrà: 45;

se si scrive:

PRINT ABS (+22)

si otterrà: 22.

Un'altra utile funzione di libreria è la funzione **SGN** (X), la quale fornisce il valore 1 se X>Ø; il valore Ø se X=Ø ed il valore –1 se X<Ø.

Funzione SGN

Per esempio, scrivendo:

PRINT SGN (5)

si otterrà: 1;

scrivendo:

PRINT SGN (Ø)

si otterrà: Ø;

scrivendo:

PRINT SGN (–7)

si otterrà: –1.

Le funzioni matematiche del BASIC sinora studiate sono riassunte di seguito.

SIN (X)	per calcolare il seno di x;
COS (X)	per calcolare il coseno di x;
TAN (X)	per calcolare la tangente di x;
ATN (X)	per calcolare l'arcotangente di x;
EXP (X)	per calcolare e^x;
LOG (X)	per calcolare il logaritmo in base e di x;
ABS (X)	per calcolare il valore assoluto di x;
SQR (X)	per calcolare la radice quadrata di x;
INT (X)	per calcolare il valore intero di x;
SGN (X)	che assume i valori 1 se x>Ø, Ø se x=Ø, –1 se x<Ø.

La forma di scrittura di ciascuna funzione elencata è costituita da tre lettere dell'alfabeto (che danno il nome alla funzione stessa), seguite da un

numero (argomento) su cui essa deve operare racchiuso fra parentesi.

2.1 – L'istruzione DEF FN

Come precisato nella precedente lezione, una funzione matematica è definita da una qualsiasi espressione algebrica che lega tra loro una o piú variabili, alcune delle quali si definiscono **variabili indipendenti**, ed altre **variabili dipendenti**.

La dipendenza tra variabili dipendenti ed indipendenti è espressa appunto da una funzione o espressione matematica.

Nell'espressione matematica sono riportate soltanto le variabili indipendenti i cui valori possono variare in modo arbitrario ed in corrispondenza degli stessi avremo altrettanti valori delle variabili dipendenti.

Per esempio, l'espressione seguente:

$$1)\ Y(X)=X^2-1$$

indica una funzione denominata Y(X), che varia in maniera dipendente dai valori arbitrari assegnati alla variabile X.

Geometricamente la funzione 1) rappresenta una parabola con il vertice nel punto Po e coordinate Ø e –1, passante per i punti dell'asse X:

$$X1=+1$$
$$X2=-1.$$

Il suo andamento è rappresentato nella *fig. 7*.

Si è già detto che alcune funzioni matematiche sono già programmate nel personal computer, per cui per eseguirle è sufficiente richiamarle.

Per il calcolo di qualsiasi altra funzione matematica definita dal programmatore occorre invece scrivere nel programma BASIC la forma esplicita della stessa funzione e definirla con un nome. Quest'ultimo serve come riferimento per i calcoli successivi.

La defininizione di una funzione si effettua all'inizio del programma, cioè prima che essa venga eseguita nei calcoli.

Per esempio, per calcolare la precedente funzione di secondo grado:

$$Y(X)=X^2-1$$

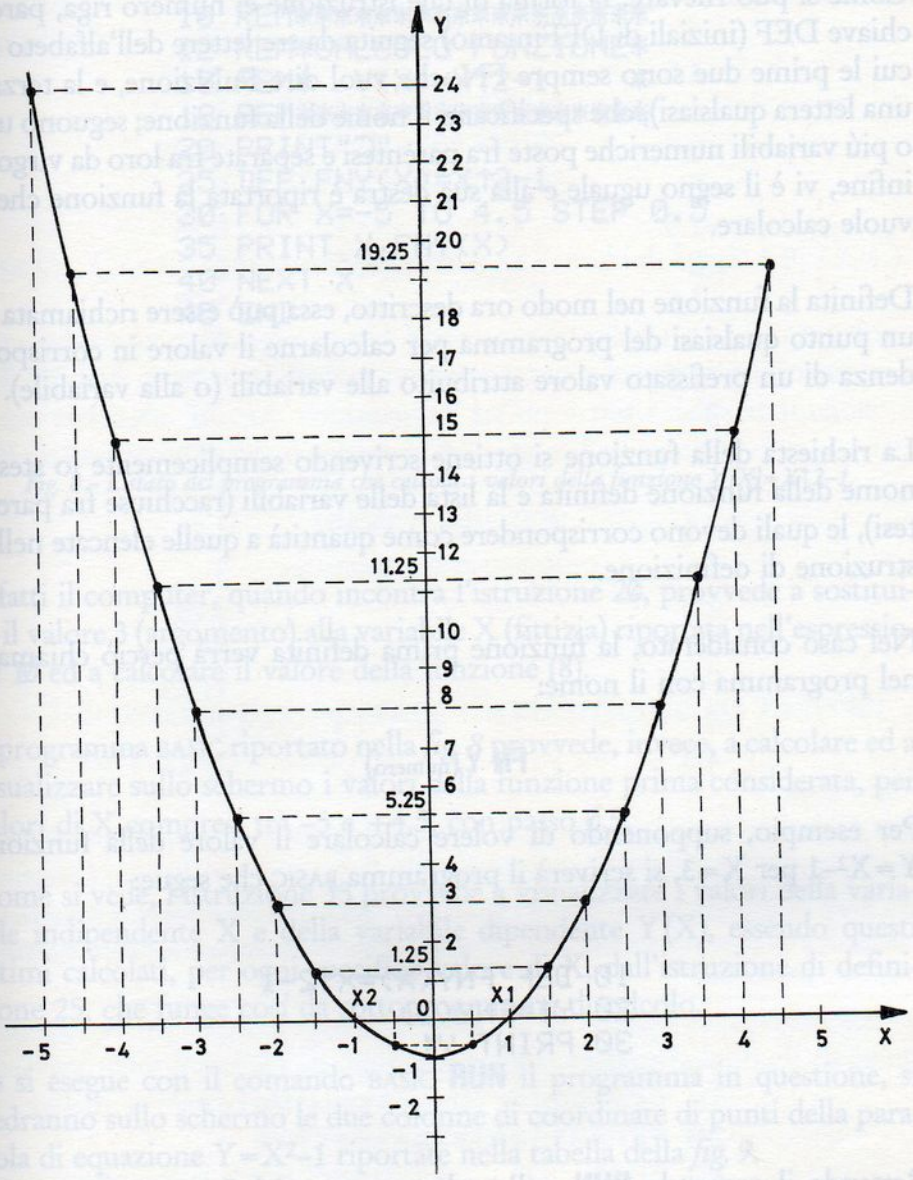
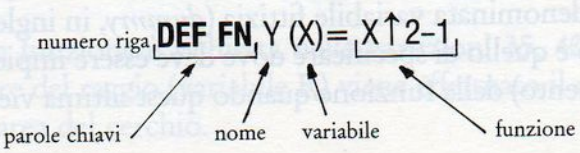


Fig. 7 - Rappresentazione geometrica della funzione parabola di equazione $Y(X)=X^2-1$.

nel programma BASIC che eseguirà questo calcolo occorrerà scrivere l'istruzione di definizione di seguito riportata:



Come si può rilevare, la forma di tale istruzione è: numero riga, parola chiave DEF (iniziali di DEFiniamo) seguita da tre lettere dell'alfabeto (di cui le prime due sono sempre FN, che vuol dire FuNzione, e la terza è una lettera qualsiasi), che specificano il nome della funzione; seguono una o piú variabili numeriche poste fra parentesi e separate fra loro da virgole; infine, vi è il segno uguale e alla sua destra è riportata la funzione che si vuole calcolare.

Definita la funzione nel modo ora descritto, essa può essere richiamata in un punto qualsiasi del programma per calcolarne il valore in corrispondenza di un prefissato valore attribuito alle variabili (o alla variabile).

La richiesta della funzione si ottiene scrivendo semplicemente lo stesso nome della funzione definita e la lista delle variabili (racchiuse fra parentesi), le quali devono corrispondere come quantità a quelle elencate nell'istruzione di definizione.

Nel caso considerato, la funzione prima definita verrà perciò chiamata nel programma con il nome:

FN Y (numero)

Per esempio, supponendo di volere calcolare il valore della funzione $Y=X^2-1$ per $X=3$, si scriverà il programma BASIC che segue:

```
10 DEF FNY(X)=X↑2-1
20 Y1=FNY(3)
30 PRINT Y1
```

Battendo il comando **RUN**, sullo schermo apparirà il numero: 8, che è appunto il valore della funzione definita nell'istruzione 1Ø per $X=3$.

Infatti, $3\uparrow 2-1=9-1=8$.

In pratica, l'istruzione 1Ø di definizione funge da subroutine per il calcolo automatico del valore della funzione Y per un certo valore di X.

La X viene denominata variabile fittizia (*dummy*, in inglese), in quanto il suo compito è quello di specificare dove deve essere impiegata la variabile reale (argomento) della funzione quando quest'ultima viene chiamata nel programma.

```
10 REM********************
12 REM*CALCOLO FUNZIONE*
15 REM*  Y(X)=X↑2-1     *
18 REM********************
20 PRINT"□"
25 DEF FNY(X)=X↑2-1
30 FOR X=-5 TO 4.5 STEP 0.5
35 PRINT X,FNY(X)
40 NEXT X
45 END
```

Fig. 8 – Listato del programma che calcola i valori della funzione Y(X)=X↑2–1.

Infatti il computer, quando incontra l'istruzione 2Ø, provvede a sostituire il valore 3 (argomento) alla variabile X (fittizia) riportata nell'espressione 1Ø ed a calcolare il valore della funzione (8).

Il programma BASIC riportato nella *fig. 8* provvede, invece, a calcolare ed a visualizzare sullo schermo i valori della funzione prima considerata, per valori di X compresi tra –5 e +4.5, con passo Ø.5.

Come si vede, l'istruzione 35 provvede a visualizzare i valori della variabile indipendente X e della variabile dipendente Y(X), essendo questi ultimi calcolati, per ogni specifico valore di X, dall'istruzione di definizione 25, che funge cosí da sottoprogramma di calcolo.

Se si esegue con il comando BASIC **RUN** il programma in questione, si vedranno sullo schermo le due colonne di coordinate di punti della parabola di equazione $Y=X^2-1$ riportate nella tabella della *fig. 9*.

Come ulteriore esempio d'impiego dell'istruzione **DEF FN** si può scrivere il programma riportato nella *fig. 10*, il quale calcola l'area del cerchio per tre diversi valori del raggio: 6.8; 7.2; 4.3.

In questo programma l'istruzione 1Ø definisce la funzione relativa al calcolo dell'area del cerchio, calcolo che viene eseguito ogni qualvolta è richiesto dal programma stesso.

In particolare, la funzione è richiesta nelle istruzioni 35, 4Ø e 45, dove per ciascun valore del raggio (variabile R) viene effettuato il calcolo della corrispondente area del cerchio.

-5	24
-4.5	19.25
-4	15
-3.5	11.25
-3	8
-2.5	5.25
-2	3
-1.5	1.25
-1	0
-0.5	-.75
0	-1
.5	-7.5
1	0
1.5	1.25
2	3
2.5	5.25
3	8
3.5	11.25
4	15
4.5	19.25

Fig. 9 – Coordinate di punti della parabola di equazione $Y=X^2-1$; la colonna di sinistra è relativa ai valori di X, quella di destra ai valori di Y.

NOTA: i valori di X sono crescenti linearmente dal valore –5 al valore 4.5, mentre quelli di Y sono prima decrescenti e poi nuovamente crescenti, con andamento detto appunto **parabolico**.

```
10 REM********************
15 REM*AREA DEL CERCHIO*
20 REM********************
25 PRINT"□"
30 DEF FNA(R)=3.14159265*R↑2
35 C1=FNA(6.8)
40 C2=FNA(7.2)
45 C3=FNA(4.3)
50 PRINT"AREA C1=";C1
55 PRINT
60 PRINT"AREA C2=";C2
65 PRINT
70 PRINT"AREA C3=";C3
75 END
```

Fig. 10 – Listato del programma che calcola l'area del cerchio per tre diversi valori del raggio.

Eseguendo il programma della *fig. 10* con il comando **RUN**, sullo schermo comparirà:

```
AREA C1=145.267244
AREA C2=162.860163
AREA C3=58.0880482
```

dove C1, C2 e C3 indicano rispettivamente i cerchi di raggio 6.8; 7.2; 4.3.

A conclusione dell'argomento è bene ancora precisare che l'istruzione **DEF FN deve sempre precedere nel programma le istruzioni in cui essa viene richiesta.** In caso contrario, si ha un arresto dell'elaborazione e sullo schermo compare il seguente messaggio d'errore:

```
UNDEF'D FUNCTION
```
(FUNZIONE NON DEFINITA)

3. - Istruzioni per operare direttamente su matrici e vettori

Come già precedentemente specificato, quasi tutti i linguaggi BASIC prevedono una dimensione massima per i componenti dei vettori o matrici, dimensione che va precisata con l'istruzione **DIM**.

Quando viene eseguita questa istruzione, il sistema BASIC provvede a riservare tante locazioni di memoria principale quanti sono gli elementi del vettore o matrice da memorizzare.

Per esempio, se si scrive l'istruzione:

```
10 DIM A (5,20), B (32)
```

il sistema provvederà a riservare $(5+1)\times(20+1)=126$ locazioni di memoria per memorizzare gli elementi della matrice A e 33 locazioni di memoria per il vettore B.

Come è noto, le dimensioni sono una in piú rispetto al numero che si dichiara nell'istruzione **DIM**, perché il sistema considera implicitamente anche gli elementi ad indice Ø (zero) di ogni vettore o matrice, cioè gli elementi A(Ø,Ø); B(Ø); C(Ø,Ø,Ø); ecc.

Occorre inoltre ricordare che gli elementi di un vettore o matrice devono essere di tipo omogeneo: o tutti valori numerici o tutti valori stringa.

Non sono ammesse forme promiscue come quelle riportate nell'esempio di matrice seguente:

$$A=\begin{pmatrix} 1 & \text{"1A00B"} \\ 2 & \text{"2A00B"} \end{pmatrix}$$

dove si sono mischiati elementi numerici con stringhe.

Le funzioni stringa, che saranno analizzate nella prossima lezione, d'altra parte consentono di effettuare eventuali separazioni fra parte numerica e parte alfanumerica di un elemento stringa. Di conseguenza, si può memorizzare in memoria la seguente matrice omogenea:

$$A\$=\begin{pmatrix} \text{"A001"} & \text{"B002"} \\ \text{"A003"} & \text{"B004"} \end{pmatrix}$$

costituita da "soli" elementi stringa e successivamente ricavare da essa un'altra matrice B (omogenea e di tipo numerico), cioè:

$$B=\begin{pmatrix} 001 & 002 \\ 003 & 004 \end{pmatrix}$$

la quale si ottiene dalla precedente separando da ogni elemento stringa le ultime tre cifre numeriche (mediante l'impiego, come si vedrà nella prossima lezione, delle funzioni stringa **VAL** e **MID$**).

La tecnica di separazione viene utilizzata spesso; per esempio, quando si deve ricavare la parte numerica da un codice di magazzino, affinché possa essere utilizzata per calcoli numerici amministrativi oltre che per identificare le parti a magazzino.

Per quanto riguarda l'istruzione **DIM** di una tabella (vettore o matrice), occorre ricordare che non è possibile dimensionare due volte una stessa tabella; soltanto in alcuni sistemi ciò è consentito, previo azzeramento della tabella stessa.

Tale azzeramento può essere effettuato attraverso particolari istruzioni BASIC (come le istruzioni **CLEAR**, **DELETE**, ecc.).

Sempre restando nel campo del calcolo matriciale, alcuni linguaggi BASIC (quelli piú evoluti) consentono l'effettuazione di operazioni con le matrici usate in matematica, come la somma e la differenza di due matrici, il prodotto di due matrici, l'inversione di una matrice, ecc., che ora passeremo in rapida rassegna.

Per sommare due matrici, queste devono avere lo stesso numero di righe e di colonne, ossia le medesime dimensioni.

Per esempio, si può scrivere in BASIC:

```
MAT C=A+B
```

il che significa che la matrice C conterrà il risultato dato dalla somma delle matrici A e B.

Analogamente l'istruzione:

```
MAT C=A-B
```

provvede a sottrarre la matrice B dalla matrice A ed a memorizzare il risultato nella matrice C.

Per moltiplicare fra loro due matrici, occorre che il numero di colonne della prima matrice sia eguale al numero di righe della seconda matrice.

Il prodotto di due matrici viene eseguito con la seguente istruzione BASIC:

```
MAT C=A*B
```

La matrice C, che conterrà il risultato del prodotto A per B, avrà lo stesso numero di righe della prima matrice (A) ed il numero di colonne della seconda matrice (B).

Queste operazioni sulle matrici, che in un normale linguaggio BASIC comportano la formulazione di un vero e proprio programma contenente non poche istruzioni, in linguaggi BASIC piú evoluti vengono realizzate, come si è visto, con semplici istruzioni (macro-istruzioni).

I dialetti del BASIC che trattano le matrici prevedono sovente anche istruzioni per calcolare la matrice inversa, la matrice trasposta, il prodotto di una matrice per una variabile, la costruzione di una matrice con tutti gli

elementi eguali a 1, ecc.

Per esempio, le istruzioni:

```
MAT A=INV (B)
MAT A=TRN (B)
```

specificano rispettivamente l'inversione (**INV**) e la trasposizione (**TRN**) della matrice B per ottenere la matrice A.

In particolare, si ottiene la trasposizione di una matrice scambiandone le righe con le colonne.

A conclusione di questa breve ed incompleta rassegna, si possono ancora citare le seguenti istruzioni riguardanti l'uso delle matrici:

```
MAT INPUT A
MAT READ A
MAT PRINT A
```

Queste istruzioni hanno lo stesso significato di **INPUT**, **READ** e **PRINT**, ma operano su matrici (o vettori).

L'istruzione **MAT INPUT** A prevede l'immissione da tastiera di tutti gli elementi costituenti la matrice A, riga dopo riga.

L'istruzione **MAT READ** A comanda la lettura, riga dopo riga, della matrice A, della quale sono state specificate le dimensioni con l'istruzione **DIM** e gli elementi con l'istruzione **DATA**.

Infine, l'istruzione **MAT PRINT** A ordina la stampa, riga per riga, della matrice A.

* * * * * * * *

In seguito...

Nella prossima lezione verranno trattate particolari funzioni del computer, in grado di operare logicamente sulle stringhe di caratteri: le funzioni di stringa sono molto utilizzate nei programmi di carattere gestionale-amministrativo.

Da memorizzare

- Quando dal programma principale si vuole saltare al sottoprogramma, si usa l'istruzione **GOSUB** seguita dal numero della prima linea del sottoprogramma. Quest'ultimo ritornerà automaticamente, attraverso l'istruzione **RETURN**, nel programma principale all'istruzione successiva alla **GOSUB**.

- **Possono essere scritte piú subroutine in uno stesso programma principale** ed ogni subroutine può saltare ad altre subroutine secondo uno schema nidificato.

- Le subroutine sono di grande utilità nei calcoli matematici e nelle elaborazioni di tipo gestionale-amministrativo. Esse consentono, inoltre, di evitare numerose ripetizioni di istruzioni, rendendo in tal modo il programma di elaborazione piú modulare nel suo complesso.

- Vengono definite **funzioni matematiche di libreria** le serie di funzioni matematiche già programmate nel computer, che possono essere immediatamente richiamate con appropriati nomi.

- Le funzioni matematiche di libreria BASIC sono: **SIN**(X), **COS**(X), **TAN**(X), **ATN**(X), **EXP**(X), **LOG**(X), **ABS**(X), **SQR**(X), **INT**(X) e **SGN**(X).

- L'istruzione **DEF FN** consente di definire qualsiasi funzione matematica precisata dal programmatore. La definizione della funzione si effettua **all'inizio del programma**, cioè prima che la funzione venga eseguita nei calcoli. In caso contrario, si ha un arresto dell'elaborazione e sullo schermo compare il messaggio d'errore: UNDEF'D FUNCTION (funzione non definita).

- I linguaggi BASIC piú evoluti permettono di **operare direttamente sulle matrici e sui vettori**, consentendo cosí di scrivere programmi piú brevi e semplici.

- Le operazioni con le matrici, previste da alcuni linguaggi BASIC, sono: la somma e la sottrazione di due matrici (**MAT** C=A+B; **MAT** C=A-B), il prodotto di due matrici (**MAT** C=A*B), il calcolo della matrice inversa (**MAT** A=**INV** (B)) e della matrice trasposta (**MAT** A=**TRN** (B)), ecc.

- Per eseguire la somma e la sottrazione fra due matrici, queste **devono avere le stesse dimensioni**. Per eseguire il prodotto di due matrici il **numero delle colonne** della prima matrice deve essere uguale al **numero delle righe** della seconda matrice.

Scuola Radio Elettra
Torino

Pratica del BASIC

Lezione 10

1 — La cassetta programmi dimostrativi n. 3
2 — Esercizio di programmazione con l'uso di sottoprogrammi
3 — Musica con il computer
4 — Suoni ed effetti sonori con il C64
5 — Sprite con il C64
6 — Suoni ed effetti sonori con il VIC-20
7 — Programma d'animazione con il VIC-20

Che cosa farà

In questa lezione eseguirà un interessante programma esemplificativo sull'uso dei sottoprogrammi.

Quindi inizierà ad esaminare la possibilità offerta dal computer di eseguire brani musicali e di generare effetti sonori particolari.

Potrà cosí apprendere gli elementi fondamentali della programmazione musicale, per la quale non è necessario essere musicisti, ma è sufficiente conoscere le note musicali.

Nell'ultima parte della lezione, dedicata alla grafica, realizzerà semplici effetti di animazione.

1 — La cassetta programmi dimostrativi n. 3

Con questo gruppo di lezioni ha ricevuto la terza cassetta, sul cui lato A sono registrati tre distinti programmi dimostrativi in linguaggio BASIC.

Il primo programma consente di giocare con le slot-machine; il secondo riguarda il calcolo di una regressione lineare, che trova applicazione nei calcoli statistici; infine, il terzo programma permette di estrarre un qualsiasi numero di valore casuale compreso tra 1 e 90, utilizzabile per formulare ambi, terne, quaterne e cinquine nel gioco del lotto.

Prima di caricare questi tre programmi nel computer ed eseguirli, è opportuno descriverli brevemente.

PROGRAMMA 1: SLOT-MACHINE

Questo programma simula il funzionamento delle slot-machine, i noti apparecchi a moneta o a gettoni installati nelle case da gioco o nei locali pubblici.

Il funzionamento di tali apparecchi è molto semplice: introducendo monete o gettoni nell'apposita feritoia, si ottiene la rotazione di tre tamburi rotanti, sui quali sono riportati diverse figure o simboli.

Al verificarsi di determinate combinazioni di simboli o figure, si ottiene una vincita il cui valore è pari ad un certo numero di volte la posta giocata.

Nel nostro caso, all'inizio del gioco il computer chiederà quanti dollari si hanno a disposizione per giocare.

Questo importo non deve essere superiore a 1ØØØ dollari; in caso contrario il computer avvertirà che la somma è troppo alta.

Qualora non si disponesse di denaro per giocare (ossia si battesse Ø), il computer generosamente "impresterà" per questo scopo 5Ø $.

La vincita si verificherà quando, ad ogni puntata, compariranno due simboli eguali oppure tre simboli eguali nelle apposite tre finestrelle della slot-machine visualizzata sullo schermo in forma stilizzata.

Con la comparsa di tre simboli uguali si ottiene ovviamente la massima vincita della giocata (en plein).

I simboli adottati dalla slot-machine Commodore sono riportati di seguito:

♥, ♠, ♦, ♣, ●, ■, X, £, +, $

Il gioco avrà termine, naturalmente, quando comparirà la scritta:

HAI PERSO TUTTO.

Per caricare il programma in questione nella memoria del computer deve battere il comando:

LOAD "SLOT"

seguito da **RETURN** .

A caricamento avvenuto, può passare ad eseguire il programma scrivendo, come al solito, il comando BASIC:

RUN

PROGRAMMA 2: REGRESSIONE LINEARE

Questo programma consente di approssimare con una retta una serie di valori numerici.

Come si vede nella *fig. 1*, i punti riportati nel grafico rappresentano valori numerici che sono "quasi" allineati lungo una retta.

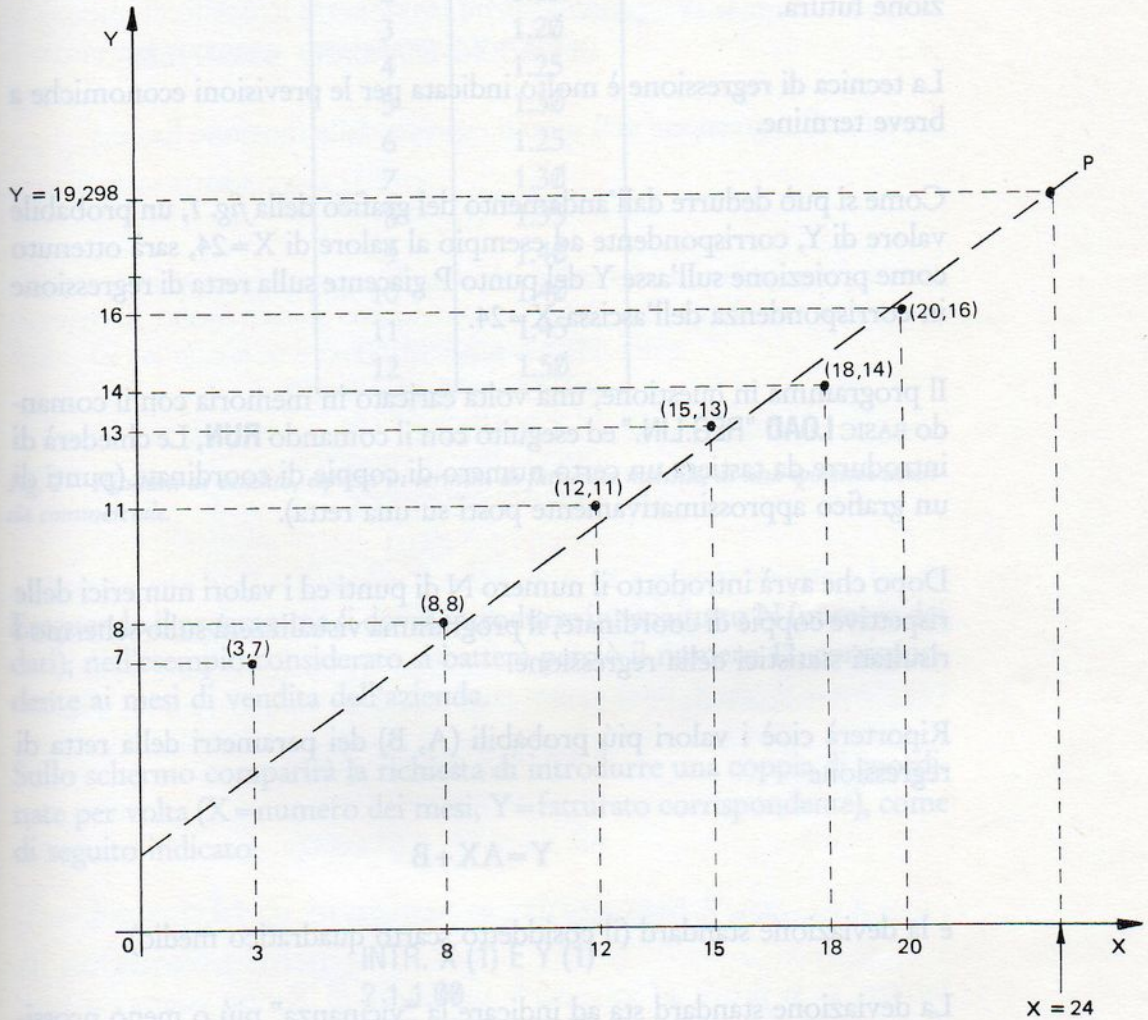


Fig. 1 – Grafico di una serie di valori numerici rappresentati tramite sei coppie di coordinate (X, Y).

I valori numerici (o punti grafici) potrebbero rappresentare i risultati fisici di un esperimento, le vendite mensili di un'azienda durante un certo periodo, le misurazioni di un fenomeno in genere.

Si tratta di determinare la retta geometricamente "piú vicina" alla serie dei valori numerici e tale determinazione si effettua matematicamente tramite il **metodo dei minimi quadrati.**

Nel grafico la retta di regressione è indicata con una linea a tratto lungo.

A che cosa serve una regressione lineare?

Principalmente a dedurre da una serie storica di dati una probabile proiezione futura.

La tecnica di regressione è molto indicata per le previsioni economiche a breve termine.

Come si può dedurre dall'andamento del grafico della *fig. 1*, un probabile valore di Y, corrispondente ad esempio al valore di X=24, sarà ottenuto come proiezione sull'asse Y del punto P giacente sulla retta di regressione in corrispondenza dell'ascissa X=24.

Il programma in questione, una volta caricato in memoria con il comando BASIC **LOAD** "REG.LIN." ed eseguito con il comando **RUN**, Le chiederà di introdurre da tastiera un certo numero di coppie di coordinate (punti di un grafico approssimativamente posti su una retta).

Dopo che avrà introdotto il numero N di punti ed i valori numerici delle rispettive coppie di coordinate, il programma visualizzerà sullo schermo i risultati statistici della regressione.

Riporterà cioè i valori piú probabili (A, B) dei parametri della retta di regressione

$$Y=AX+B$$

e la deviazione standard (il cosiddetto scarto quadratico medio).

La deviazione standard sta ad indicare la "vicinanza" piú o meno prossima della serie di punti alla retta di regressione lineare.

Oltre ai valori statistici citati prima, il programma consente di calcolare (o interpolare) la serie di punti con diversi valori di X (esterni o interni all'intervallo di definizione).

Un esempio pratico servirà a chiarire meglio quanto finora esposto.

Supponiamo che un'azienda abbia conseguito in un anno i risultati di vendita (espressi in termini di fatturato mensile) riportati nella *fig. 2*.

MESE	FATTURATO IN MILIONI
1	1.ØØ
2	1.1Ø
3	1.2Ø
4	1.25
5	1.3Ø
6	1.25
7	1.3Ø
8	1.35
9	1.4Ø
10	1.4Ø
11	1.45
12	1.5Ø

Fig. 2 – Risultati di vendita, espressi in termini di fatturato mensile, di una ipotetica azienda commerciale.

Eseguendo il programma si dovrà introdurre innanzitutto N (numero dei dati); nell'esempio considerato si batterà perciò il numero 12, corrispondente ai mesi di vendita dell'azienda.

Sullo schermo comparirà la richiesta di introdurre una coppia di coordinate per volta (X=numero dei mesi, Y=fatturato corrispondente), come di seguito indicato:

```
INTR. X (1) E Y (1)
? 1,1.ØØ
INTR. X (2) E Y (2)
? 2,1.1Ø
INTR. X (3) E Y (3)
? 3,1.2Ø
⋮                    ⋮
INTR. X (12) E Y (12)
? 12,1.5Ø
```

Dopo la battitura dell'ultima coppia di coordinate, sullo schermo compariranno i risultati della regressione con il seguente messaggio:

```
COEFFICIENTI A E B
DELLA RETTA DI REGRESSIONE
Y=AX+B
A=.Ø384615381
B=1.Ø4166667
-------------------------------

DEVIAZIONE STANDARD .Ø383277588
```

La retta di regressione sarà quindi espressa dalla funzione lineare:

$$Y=.Ø384615381X+1.Ø4166667$$

Battendo quindi un tasto, sullo schermo comparirà la scritta:

```
PROVA ADESSO AD
INTRODURRE UN VALORE
DI X COMPRESO TRA
QUELLI GIA' INTRODOTTI
OPPURE SUPERIORE
```

Se, ad esempio, si prova ad introdurre il valore X=13, il che equivale a chiedersi quanto sarà il fatturato (valore di Y) dell'azienda considerata nel mese successivo al 12°, cioè in quello che segue l'ultimo mese della serie riportata nella *fig. 2*, il computer risponderà:

```
IL VALORE PRESUMIBILE
DI Y CORRISPONDENTE
AD X=13 E' : 1.54166666
```

Ossia il fatturato presumibile sarà di 1.541.666.

Inoltre, se si prova ad introdurre un valore di X non compreso fra i dodici valori interi elencati nella tabella della *fig. 2*, ma comunque maggiore di 1 e minore di 12, ad esempio 9.5, si avrà un valore di Y=1.4Ø7Ø5128.

Questo valore rappresenta la presumibile vendita che si sarebbe potuta realizzare a metà settembre.

Vediamo infine che cosa succede se si prova ad introdurre per X uno dei 12 valori interi, già presenti in memoria (1, 2, 3, ... 12); ad esempio, per X=2 (febbraio) la risposta sarebbe: Y=1.11858975, un po' diversa da quella realizzata effettivamente nel mese di febbraio (Y=1.1Ø).

Il motivo di questa differenza risiede nel fatto che i valori di Y sono "teorici", cioè basati sul presupposto che l'andamento di base del fenomeno di vendite sia rigorosamente lineare (dal primo all'ultimo mese dell'anno).

In pratica, l'utilizzo di presupposti probabilistici porta sempre a margini d'errore nel riscontro con la realtà.

PROGRAMMA 3: GIOCO DEL LOTTO

Questo programma consente di estrarre, come già accennato, un qualsiasi numero di valore casuale compreso fra 1 e 9Ø, utilizzabile per formulare ambi, terne, quaterne e cinquine nel gioco del lotto.

Per caricare tale programma nel computer, occorre eseguire il comando BASIC:

LOAD "LOTTO"

Impartito il comando d'esecuzione **RUN**, il programma, prima di estrarre il numero a caso, chiede all'operatore esterno se intende continuare ad estrarre numeri oppure se vuole interrompere l'operazione. A questa domanda si dovrà rispondere con un SI oppure con un NO.

2. - Esercizio di programmazione con l'uso di sottoprogrammi

I programmi complessi, nei quali si verifica spesso la necessità di dover ripetere piú volte la stessa sequenza di operazioni, possono essere costruiti a guisa di moduli separati, cioè in gruppi di sottoprogrammi.

Questa tecnica "modulare", come abbiamo visto nella lezione *Analisi e programmazione 10*, serve principalmente per una migliore comprensione di un programma: infatti, nel caso in cui dovesse verificarsi un errore durante l'elaborazione, sarebbe oltremodo agevole individuare la parte (o modulo) di programma in cui è presente l'errore stesso.

L'esercizio di programmazione che stiamo per formulare, consistente in un gioco tipo indovinello, è un chiaro esempio di utilizzazione dei sottoprogrammi.

Il programma in questione permette di indovinare un numero che sia stato precedentemente estratto dal calcolatore (tramite la funzione matematica RND (X)).

Come è noto, mediante questa funzione il computer estrae a caso un valore numerico qualsiasi compreso tra Ø e 1, valore che, attraverso opportune operazioni algebriche, viene trasformato in un numero intero compreso tra 1 e 1ØØ.

L'operatore ha a disposizione dieci mosse per indovinare quel numero; ogni volta che egli introduce un numero tramite tastiera, il computer risponde con un "messaggio guida", che serve appunto a segnalargli la vicinanza piú o meno prossima al numero casuale.

I messaggi forniti sono otto e vanno dalle "Acque profonde"... al "Fuochino", ecc. fino ad arrivare all'"Incendio" ed infine al "CENTRATO", per significare che si è indovinato "esattamente" il numero casuale estratto dal calcolatore.

Comunque vadano le cose, cioè che si sia indovinato questo valore o che si siano esaurite tutte e dieci le mosse, il computer emana un "giudizio" finale sul gioco ad indovinello. Il giudizio consiste nel segnalare all'operatore in quante mosse è riuscito a "centrare" il bersaglio, oppure nell'avvertirlo che ha fallito la prova.

Nella *fig. 3* è riportato il flow-chart del programma in questione.

Come già visto, il simbolo grafico adottato per indicare l'esecuzione di un modulo o sottoprogramma è costituito da un rettangolo avente due barrette in corrispondenza dei lati verticali.

Nella *fig. 4* e nella *fig. 5* è riportata una possibile codifica in linguaggio BASIC del flow-chart della *fig. 3*.

Sono state inserite nel programma speciali istruzioni di commento (con la **REM**) che servono ad evidenziare meglio le subroutine indicate nel flow-chart con i blocchi 1), 2), 3), ... 6).

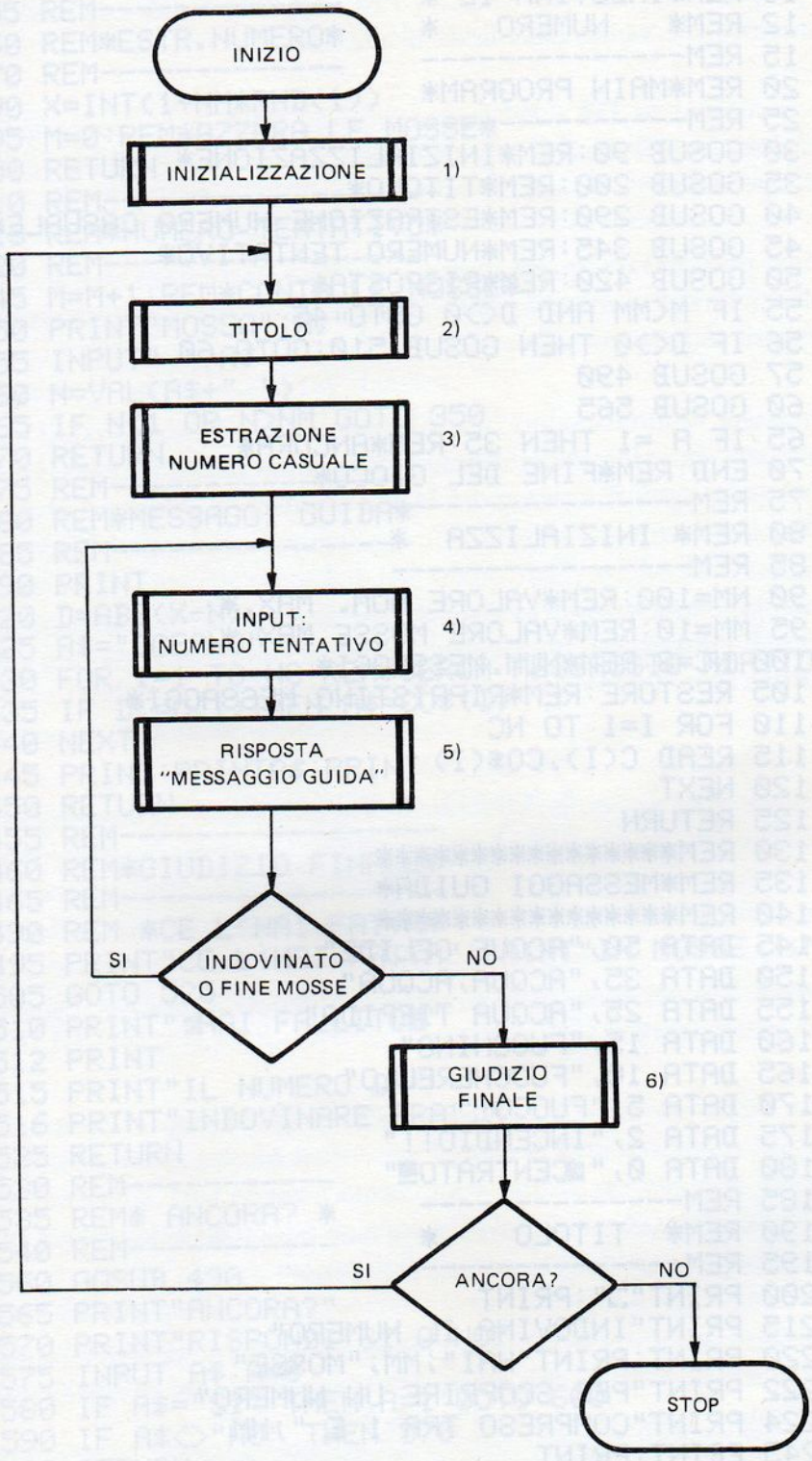


Fig. 3 – Flow-chart del programma indovinello.

```
10 REM*INDOVINA IL *
12 REM*   NUMERO   *
15 REM--------------
20 REM*MAIN PROGRAM*
25 REM--------------
30 GOSUB 90:REM*INIZIALIZZAZIONE*
35 GOSUB 200:REM*TITOLO*
40 GOSUB 290:REM*ESTRAZIONE NUMERO CASUALE*
45 GOSUB 345:REM*NUMERO TENTATIVO*
50 GOSUB 420:REM*RISPOSTA*
55 IF M<MM AND D<>0 GOTO 45
56 IF D<>0 THEN GOSUB 510:GOTO 60
57 GOSUB 490
60 GOSUB 565
65 IF A =1 THEN 35:REM*ANCORA*
70 END:REM*FINE DEL GIOCO*
75 REM----------------
80 REM* INIZIALIZZA  *
85 REM----------------
90 NM=100:REM*VALORE NUM. MAX.*
95 MM=10:REM*VALORE MOSSE MAX.*
100 NC=8:REM*NUM.MESSAGGI*
105 RESTORE:REM*RIPRISTINO MESSAGGI*
110 FOR I=1 TO NC
115 READ C(I),CO$(I)
120 NEXT
125 RETURN
130 REM*****************
135 REM*MESSAGGI GUIDA*
140 REM*****************
145 DATA 50,"ACQUE GELIDE"
150 DATA 35,"ACQUA,ACQUA"
155 DATA 25,"ACQUA TIEPIDA"
160 DATA 15,"FUOCHINO"
165 DATA 10,"FUOCHERELLO"
170 DATA 5,"FUOCO!"
175 DATA 2,"INCENDIO!!"
180 DATA 0,"■CENTRATO■"
185 REM--------------
190 REM*  TITOLO    *
195 REM--------------
200 PRINT"■":PRINT
215 PRINT"INDOVINA IL NUMERO"
220 PRINT:PRINT"HAI";MM;"MOSSE"
222 PRINT"PER SCOPRIRE UN NUMERO"
224 PRINT"COMPRESO TRA 1 E ";NM
245 PRINT:PRINT
250 RETURN
```

Fig. 4 – Listato del programma indovinello.

```
255 REM-------------
260 REM*ESTR.NUMERO*
270 REM-------------
290 X=INT(1+NM*RND(1))
295 M=0:REM*AZZERA LE MOSSE*
300 RETURN
310 REM-----------------
315 REM*NUMERO TENTATIVO*
320 REM-----------------
345 M=M+1:REM*CONTA LE MOSSE*
350 PRINT"MOSSA";M;":";
355 INPUT" ";A$
360 N=VAL(A$+" ")
365 IF N<1 OR N>NM GOTO 350
370 RETURN
375 REM----------------
380 REM*MESSAGGI GUIDA*
385 REM----------------
390 PRINT
420 D=ABS(X-N)
425 A$="OCEANO"
430 FOR I=1 TO NC:REM*CERCA MESSAGGIO ADATTO*
435 IF D<=C(I)THEN A$=CO$(I)
440 NEXT
445 PRINT:PRINTA$:PRINT
450 RETURN
455 REM-----------------
460 REM*GIUDIZIO FINALE*
465 REM-----------------
490 REM *CE L'HAI FATTA*
495 PRINT"CE L'HAI FATTA":PRINT"IN MOSSE ";M
505 GOTO 525
510 PRINT"■HAI FALLITO■"
512 PRINT
515 PRINT"IL NUMERO DA"
516 PRINT"INDOVINARE ERA";X
525 RETURN
530 REM----------
535 REM* ANCORA? *
540 REM----------
560 GOSUB 490
565 PRINT"ANCORA?"
570 PRINT"RISPONDI SI O NO"
575 INPUT A$:A=0
580 IF A$="SI" THEN A=1:GOTO 600
590 IF A$<>"NO" THEN 570
600 RETURN
```

Fig. 5 – Proseguimento del listato relativo al programma indovinello.

Il programma principale (MAIN PROGRAM) risulta cosí ridotto alle istruzioni che vanno dalla linea 1Ø alla linea 7Ø.

Le istruzioni da 13Ø a 18Ø sono istruzioni **DATA** in cui sono riportati i MESSAGGI GUIDA che il computer emana in risposta ai tentativi numerici dell'operatore.

Come può rilevare, questi messaggi guidano l'operatore esterno all'individuazione del numero da indovinare (numero che è ben nascosto nella memoria del computer!).

I messaggi vanno da valori "freddi" a valori "caldi": quanto piú il messaggio è "freddo", tanto piú il numero introdotto risulta lontano da quello vero; quanto piú il messaggio è "caldo", tanto piú il numero è vicino al vero.

Come già detto, si hanno a disposizione dieci mosse; se al termine di esse non si è indovinato il numero misterioso, il computer comunica all'operatore:

```
HAI FALLITO
IL NUMERO DA INDOVINARE
ERA: X
```

dove X sta ad indicare il numero da indovinare.

Se invece nel corso dei dieci tentativi si indovina il numero misterioso, viene emesso il messaggio:

```
CE L'HAI FATTA
IN N MOSSE
```

dove N è l'ennesimo tentativo fatto per indovinare il numero, ossia l'ennesimo numero introdotto nel computer.

Al termine del gioco, se si vuole indovinare un altro numero (compreso sempre fra 1 e 1ØØ e scelto a caso), occorre battere SI; altrimenti si interrompe il gioco battendo la parola NO.

Conclusa l'analisi del programma, Lei può immetterlo nel computer e quindi eseguirlo con il comando BASIC **RUN**.

3 — Musica con il computer

Sin dalla piú remota antichità l'uomo ha cercato di imitare i suoni della natura che gli stava intorno, determinando cosí la nascita e l'evoluzione di forme di linguaggio particolari: il canto e la musica.

Reperti archeologici testimoniano che l'uomo costruì i primi rudimentali strumenti musicali già agli albori della sua storia, utilizzando le ossa e le interiora degli animali uccisi per sfamarsi.

La prima codifica dell'arte musicale fu opera degli antichi greci; costoro, e Pitagora fu tra questi, scoprirono che la musica seguiva leggi armoniche ben precise. Sembrerà strano, ma i primi cultori dell'arte musicale codificata furono dei matematici ed il fenomeno dello studio di quest'arte rimase circoscritto a questi pochi eletti e non diventò patrimonio del popolo, cosicché ancora per molti secoli non si seppe come codificare la musica in modo da poterla tramandare ai posteri o comunicarla agli altri senza bisogno dell'insegnamento diretto.

Si deve giungere al Medio Evo per trovare i primi testi di musica scritta: il cristianesimo si era ormai diffuso ed era sorta l'esigenza, tra i monaci o i religiosi in genere, di comunicare ai confratelli i canti liturgici. Fu così che nell'Europa di quel tempo ebbe ampia diffusione il canto Gregoriano e fu proprio in quel periodo che alle note musicali fu attribuita (da Guido d'Arezzo, teorico dell'XI secolo) l'etichetta ancora oggi in vigore: UT, RE, MI, FA, SOL, LA, SI. In seguito, UT si trasformò in DO.

Per la verità, a quei tempi le conoscenze sulla fisica dei suoni erano praticamente nulle, per cui molte cose venivano fatte unicamente seguendo l'intuito ed in base all'esperienza. Quest'ultima, per esempio, insegnava che partendo da un suono, quale un DO, e variando quella che allora veniva chiamata "altezza del suono" (fu scoperto solo in seguito che si trattava della frequenza), si percorrevano tutte le note e si giungeva ad un suono molto simile a quello di partenza, tant'è vero che, se si suonavano (o cantavano) i due suoni all'unisono, essi apparivano indistinguibili.

Per questi motivi anche il secondo suono fu battezzato con la stessa etichetta del primo (DO, nel nostro esempio) e, poiché tra il primo ed il secondo c'erano di mezzo otto suoni, l'intervallo fu chiamato **ottava**.

Solamente in epoca recente si scoprì che, passando da un DO a quello successivo, la frequenza del secondo è esattamente il doppio di quella del primo.

Ben presto fu chiaro che, per comporre musica, sette note erano poche: l'intervallo di un'ottava veniva diviso nella pratica in piú di otto parti (otto suoni), senza però che ciò fosse codificato. Questa situazione conduceva al fatto che ogni esecutore, piú che "eseguire" un brano, lo "interpretava" a propria discrezione, con l'evidente risultato di aggiungere nuova confusione a quella che già esisteva.

Si deve a J.S. Bach un intervento deciso e risolutore sul modo di scrivere la musica.

Egli suddivise l'intervallo di una ottava in dodici intervalli esatti e, senza allontanarsi dalla tradizione musicale, battezzò le note comprese in questo intervallo nel modo tuttora adottato: DO, DO # (DO diesis), RE, RE #, MI, FA, FA #, SOL, SOL #, LA, LA #, SI.

Alla luce delle conoscenze della fisica moderna si può affermare che le note si distinguono essenzialmente, anche se non esclusivamente, dalla loro frequenza. Due note la cui frequenza è una il doppio dell'altra portano la stessa etichetta e l'intervallo è detto ottava.

Come punto di riferimento assoluto si assume il LA della quarta ottava di un pianoforte la cui frequenza è di 440 Hz. Quindi, in base a quanto detto, il LA dell'ottava superiore avrà una frequenza di 880 Hz mentre il LA dell'ottava inferiore sarà di 220 Hz.

L'intervallo di un'ottava è diviso in dodici parti (dodicinale), in modo che il rapporto tra le frequenze di due note consecutive si mantenga costante. Questo rapporto vale $\sqrt[12]{2} = 1{,}05946$. Pertanto, il LA # successivo al LA di riferimento ha una frequenza pari a:

$$440 \times 1{,}05946 = 466{,}164 \text{ Hz (LA \#)}.$$

Analogamente il SI successivo ha la frequenza di:

$$466{,}164 \times 1{,}05946 = 493{,}883 \text{ Hz (SI)}.$$

Proseguendo in questo modo per le note succcessive, si arriva al LA dell'ottava superiore, la cui frequenza è di 880 Hz.

Analizzando le note generate dai vari strumenti musicali, si può rilevare che, mentre la frequenza si mantiene costante per ogni nota, uno degli elementi che varia da uno strumento all'altro è la **forma d'onda.** Questa è comunemente chiamata **timbrica** ed è appunto in base al timbro che si distingue un LA suonato da un flauto da un LA suonato da un clarinetto. Senza addentrarci troppo in giustificazioni teoriche, si può affermare che ogni strumento opera come un filtro selettivo quando genera una certa nota ed ogni tipo di strumento si comporta come un filtro diverso.

Un terzo elemento che caratterizza gli strumenti musicali è la **variazione istantanea dell'intensità del suono** nell'ambito di ogni singola nota.

Questa variazione di intensità è detta **inviluppo.** Ad esempio, quando si preme un tasto di un pianoforte per suonare una nota, l'intensità del suono passa rapidamente **dal valore zero** (silenzio) **al valore massimo,** per poi scendere dolcemente fino ad arrivare ad **un valore,** detto di **sostentamento,** che si mantiene per un intervallo di tempo relativamente lungo; infine, il suono si smorza completamente. Nella *fig. 6* è indicato questo tipico andamento, caratteristico di molti strumenti a corda.

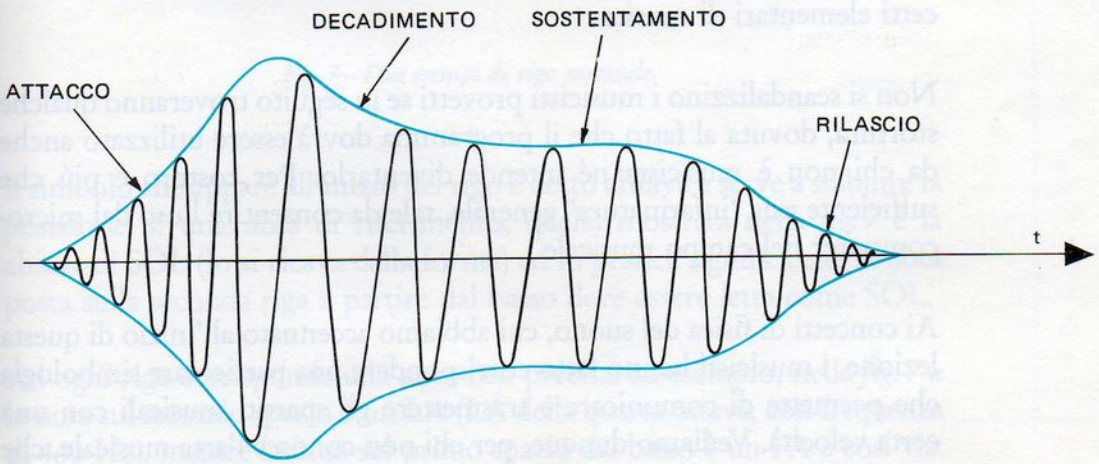


Fig. 6 – Andamento del suono di una nota generata da un pianoforte.

Naturalmente strumenti di tipo diverso, come quelli a percussione oppure a fiato, hanno forme di inviluppo differenti da quella descritta prima.

Negli anni piú recenti, con lo sviluppo delle tecniche elettroniche e quindi con la capacità di generare forme d'onda di qualsiasi tipo, si è cercato di realizzare, mediante circuiti elettronici, apparecchi capaci di imitare il comportamento degli strumenti musicali.

Sono cosí nati dapprima gli organi elettronici ed in seguito i cosiddetti "sintetizzatori", i quali non solo imitano gli strumenti veri, ma sono capaci persino di "inventare" strumenti nuovi dal suono "innaturale".

Naturalmente, i risultati conseguiti con gli strumenti elettronici hanno fatto e fanno tuttora storcere il naso ai puristi, che spesso li rifiutano. Comunque, qualunque sia l'idea che si ha su questi strumenti, è un fatto che ormai sono ampiamente diffusi e la storia della musica non può piú prescindere da essi.

In questi ultimi anni, poi, la diffusione degli elaboratori elettronici ha consentito da una parte l'approfondimento dello studio dei suoni generati dagli strumenti musicali tradizionali tramite una tecnica detta **analisi del suono** e dall'altra la realizzazione di strumenti sofisticati, utili per generare nuovi suoni e nuove armonie mediante la **sintesi del suono.**

Poiché intendiamo analizzare il sistema per programmare un brano musicale con il computer valido, almeno nell'impostazione di base, sia per coloro che conoscono la musica sia per coloro che sono privi di conoscenze nel settore, affronteremo l'argomento introducendo alcuni concetti elementari di musica.

Non si scandalizzino i musicisti provetti se in seguito troveranno qualche stortura, dovuta al fatto che il programma dovrà essere utilizzato anche da chi non è musicista né intende diventarlo. Per costoro è piú che sufficiente una "infarinatura" generale, tale da consentire l'uso del microcomputer nel campo musicale.

Ai concetti di fisica del suono, cui abbiamo accennato all'inizio di questa lezione, i musicisti hanno fatto corrispondere una particolare simbologia che permette di comunicare e trasmettere gli spartiti musicali con una certa velocità. Vediamo dunque, per chi non conosce l'arte musicale, che cosa è la musica scritta.

Tutti hanno certamente sentito parlare di pentagramma; ebbene, è proprio sul pentagramma che si scrive la musica. Le note cioè vengono disegnate su un rigo musicale composto da cinque righe (da cui deriva appunto il nome "pentagramma") e da quattro spazi. Vengono utilizzati anche gli spazi immediatamente inferiore e superiore al rigo.

Nella *fig. 7* sono rappresentati due esempi di rigo musicale con alcune note.

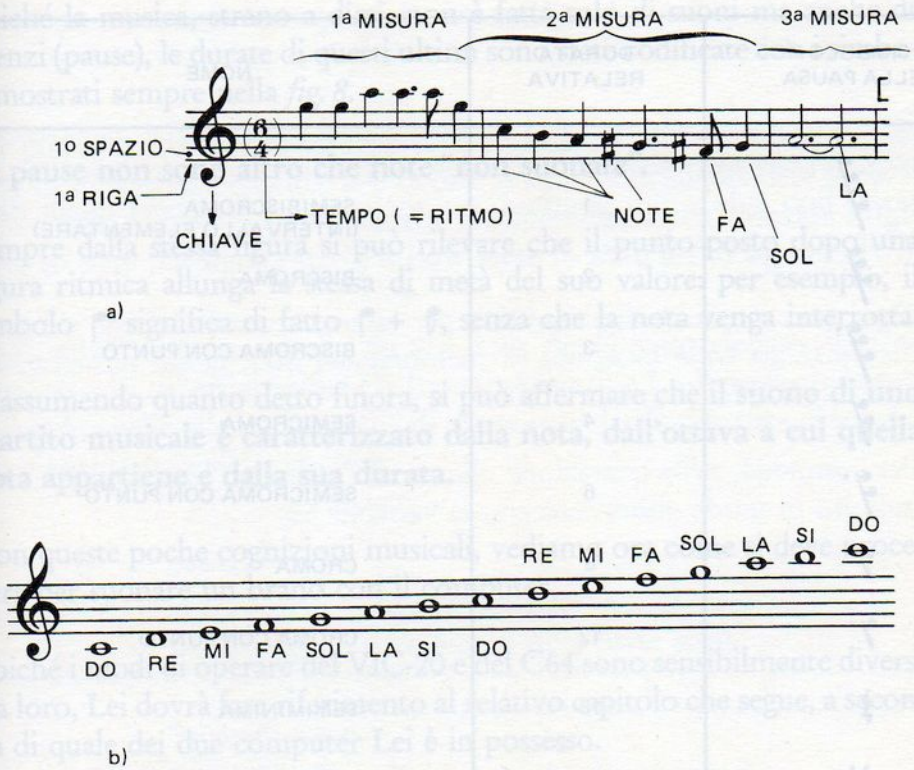


Fig. 7 – Due esempi di rigo musicale.

Il simbolo che appare all'inizio del rigo è detto **chiave** e serve a stabilire la posizione di una nota di riferimento; quella mostrata nella *fig. 7* è la chiave di SOL (lo si ricava dalla forma) ed in pratica significa che la nota posta sulla seconda riga a partire dal basso deve essere letta come SOL.

Ad ogni riga corrisponde una nota ben precisa: ad esempio, nella *fig. 7-a* la nota sul secondo spazio è un LA (LA della quarta ottava, con frequenza di 440 Hz), mentre la nota sul primo spazio dal basso è un FA e cosí via.

Nella *fig. 7-b* è riportato un rigo musicale contenente le note di due ottave.

I semitoni non hanno una posizione particolare, ma sono indicati dalla nota seguita dal simbolo # (diesis).

La musica, tuttavia, non è un semplice e banale succedersi di note: è la durata di ogni singola nota che consente di percepire quello che chiamiamo comunemente "motivo".

SIMBOLO DELLA NOTA	SIMBOLO DELLA PAUSA	DURATA RELATIVA	NOME
		1	SEMIBISCROMA (INTERVALLO ELEMENTARE)
		2	BISCROMA
		3	BISCROMA CON PUNTO
		4	SEMICROMA
		6	SEMICROMA CON PUNTO
		8	CROMA
		12	CROMA CON PUNTO
		16	SEMIMINIMA
		24	SEMIMINIMA CON PUNTO
		32	MINIMA
		48	MINIMA CON PUNTO
		64	SEMIBREVE
		96	SEMIBREVE CON PUNTO
		128	BREVE

Fig. 8 – Simboli per la durata delle note e delle pause (figure ritmiche).

La durata di una nota è espressa con i simboli indicati nella *fig. 8*, osservando la quale si rileva che tale durata non è assunta in valore assoluto, bensí relativo.

Se al primo simbolo si attribuisce durata 1, al secondo si deve attribuire un valore doppio, cioè 2, al terzo un valore doppio del precedente, cioè 4, e cosí via. La velocità di esecuzione del brano dipende dunque dalla durata che si attribuisce al valore 1, naturalmente in termini assoluti di tempo.

Poiché la musica, strano a dirsi, non è fatta solo di suoni ma anche di silenzi (pause), le durate di questi ultimi sono state codificate con i simboli mostrati sempre nella *fig. 8*.

Le pause non sono altro che note "non suonate".

Sempre dalla stessa figura si può rilevare che il punto posto dopo una figura ritmica allunga la stessa di metà del suo valore: per esempio, il simbolo 𝅗𝅥. significa di fatto 𝅗𝅥 + 𝅘𝅥, senza che la nota venga interrotta.

Riassumendo quanto detto finora, si può affermare che il **suono di uno spartito musicale è caratterizzato dalla nota, dall'ottava a cui quella nota appartiene e dalla sua durata.**

Con queste poche cognizioni musicali, vediamo ora come si deve procedere per suonare un brano con il computer.

Poiché i modi di operare del VIC-20 e del C64 sono sensibilmente diversi fra loro, Lei dovrà fare riferimento al relativo capitolo che segue, a seconda di quale dei due computer Lei è in possesso.

4 — Suoni ed effetti sonori con il C64

Con il C64 è possibile programmare musica, intendendo con il termine "programmare" il fatto che le note musicali possono essere attivate attraverso un programma BASIC.

A questo scopo il C64 è dotato di un vero e proprio sintetizzatore musicale, che utilizza un particolare circuito denominato SID (dispositivo interfaccia del suono), con cui si possono creare tre diverse tonalità musicali (o voci), con un'esecuzione di nove ottave e generare rumori per effetti sonori, note e rumori che vengono emessi dall'altoparlante dell'apparecchio televisivo collegato al computer stesso.

Ciascuna tonalità o voce è programmabile separatamente ed indipendentemente dalle altre.

Inoltre, un apposito circuito denominato ADSR (iniziali delle parole attaccare-decadere-sostenere-rilasciare) permette di ottenere, per ciascuna voce, non solo suoni continui ma anche impulsivi come quelli degli strumenti a percussione.

Ogni voce necessita di alcuni dati per essere suonata, dati che vengono memorizzati nelle locazioni di memoria del SID, la cui numerazione inizia da 54272 e termina a 54296.

Per comodità, nella trattazione che segue l'indirizzo base del SID verrà indicato in modo abbreviato con la variabile SI:

SI=54272

Le locazioni di memoria del SID si possono suddividere in quattro gruppi principali, come di seguito riportato:

da 54272 (SI)	a 54278 (SI+6)	= prima voce (o tonalità);
da 54279 (SI+7)	a 54285 (SI+13)	= seconda voce;
da 54286 (SI+14)	a 54292 (SI+20)	= terza voce;
da 54293	a 54296	= locazioni comuni alle tre voci.

Vediamo ora, molto brevemente, i principali elementi che caratterizzano un suono.

4.1 - Caratteri del suono

Come è noto, un suono è generato dalla vibrazione di un corpo elastico; tali vibrazioni si propagano in un mezzo anch'esso elastico (generalmente l'aria) sotto la forma di onde sonore.

Il motivo per cui un suono risulta piú o meno grave o acuto dipende, come già accennato, dalla sua **altezza**, ossia dalla frequenza delle vibrazioni che lo costituiscono.

Ad esempio, un suono acuto ha un numero di vibrazioni superiore a quello delle vibrazioni che contraddistinguono un suono grave.

La frequenza di una nota è determinata, nel C64, da un parametro che può assumere un valore superiore a 255.

Per potere memorizzare valori cosí elevati non è sufficiente un singolo byte di memoria. Pertanto, si è ricorsi al sistema di suddividere il parametro della frequenza in 2 byte, che chiameremo rispettivamente **High Byte** (o byte piú significativo) e **Low Byte** (o byte meno significativo).

Nella tabella riportata nella *fig. 9* e nella *fig. 10* sono elencati le note musicali ed i rispettivi valori Hi-Byte (byte alto) e Lo-Byte (byte basso) da inserire nelle relative locazioni di memoria del SID riportate di seguito per ciascuna voce del computer.

Prima voce
54272 (SI) =frequenza nota (byte basso)
54273 (SI+1) =frequenza nota (byte alto)

Seconda voce
54279 (SI+7) =frequenza nota (byte basso)
54280 (SI+8) =frequenza nota (byte alto)

Terza voce
54286 (SI+14) =frequenza nota (byte basso)
54287 (SI+15) =frequenza nota (byte alto)

Un altro particolare carattere del suono è il **timbro**, il quale è dovuto al fatto che un corpo vibrante produce, unitamente al suono fondamentale, altri suoni piú o meno acuti.

Sono appunto questi suoni armonici che danno quel particolare "colorito" al suono fondamentale, anche se questo rimane sempre il suono dominante.

Il C64 può generare tre differenti forme d'onda per la produzione dei suoni, piú una forma d'onda per la generazione dei rumori che esamineremo in una prossima lezione.

La prima forma d'onda, denominata **triangolare** (*fig. 11-a*), è quella che consente di ottenere suoni piú melodiosi.

N.	Note-Ottave	Frequenza (Hz)	Hi -Byte	Lo-Byte
0	DO-0	16.4	1	22
1	DO#-0	17.3	1	39
2	RE-0	18.4	1	57
3	RE#-0	19.4	1	75
4	MI-0	20.6	1	95
5	FA-0	21.8	1	116
6	FA#-0	23.1	1	138
7	SOL-0	24.5	1	161
8	SOL#-0	26.0	1	186
9	LA-0	27.5	1	212
10	LA#-0	29.1	1	240
11	SI-0	30.9	2	14
12	DO-1	32.7	2	45
13	DO#-1	34.6	2	78
14	RE-1	36.7	2	113
15	RE#-1	38.9	2	150
16	MI-1	41.2	2	190
17	FA-1	43.7	2	231
18	FA#-1	46.2	3	20
19	SOL-1	49.0	3	66
20	SOL#-1	51.9	3	116
21	LA-1	55.0	3	169
22	LA#-1	58.3	3	224
23	SI-1	61.7	4	27
24	DO-2	65.4	4	90
25	DO#-2	69.3	4	156
26	RE-2	73.4	4	226
27	RE#-2	77.8	5	45
28	MI-2	82.4	5	123
29	FA-2	87.3	5	207
30	FA#-2	92.5	6	39
31	SOL-2	98.0	6	133
32	SOL#-2	103.8	6	232
33	LA-2	110.0	7	81
34	LA#-2	116.5	7	193
35	SI-2	123.5	8	55
36	DO-3	130.8	8	180
37	DO#-3	138.6	9	56
38	RE-3	146.8	9	196
39	RE#-3	155.6	10	89
40	MI-3	164.8	10	247
41	FA-3	174.6	11	158
42	FA#-3	185.0	12	78
43	SOL-3	196.0	13	10
44	SOL#-3	207.7	13	208
45	LA-3	220.0	14	162
46	LA#-3	233.1	15	129
47	SI-3	246.9	16	109

Fig. 9 – Valori delle note musicali da adottare per il C64.

(segue su fig. 10)

N.	Note-Ottave	Frequenza (Hz)	Hi-Byte	Lo-Byte
48	DO-4	261.6	17	103
49	DO#-4	277.2	18	112
50	RE-4	293.7	19	137
51	RE#-4	311.1	20	178
52	MI-4	329.6	21	237
53	FA-4	349.2	23	59
54	FA#-4	370.0	24	157
55	SOL-4	392.0	26	20
56	SOL#-4	415.3	27	160
57	LA-4	440.0	29	69
58	LA#-4	466.2	31	3
59	SI-4	493.9	32	219
60	DO-5	523.3	34	207
61	DO#-5	554.4	36	225
62	RE-5	587.3	39	18
63	RE#-5	622.3	41	101
64	MI-5	659.3	43	219
65	FA-5	698.5	46	118
66	FA#-5	740.0	49	58
67	SOL-5	784.0	52	39
68	SOL#-5	830.6	55	65
69	LA-5	880.0	58	138
70	LA#-5	932.3	62	5
71	SI-5	987.8	65	181
72	DO-6	1046.5	69	157
73	DO#-6	1108.7	73	193
74	RE-6	1174.7	78	36
75	RE#-6	1244.5	82	201
76	MI-6	1318.5	87	182
77	FA-6	1396.9	92	237
78	FA#-6	1480.0	98	115
79	SOL-6	1568.0	104	78
80	SOL#-6	1661.2	110	130
81	LA-6	1760.0	117	20
82	LA#-6	1864.7	124	10
83	SI-6	1975.5	131	106
84	DO-7	2093.0	139	59
85	DO#-7	2217.5	147	130
86	RE-7	2349.3	156	72
87	RE#-7	2489.0	165	147
88	MI-7	2637.0	175	107
89	FA-7	2793.8	185	218
90	FA#-7	2960.0	196	231
91	SOL-7	3136.0	208	156
92	SOL#-7	3322.4	221	4
93	LA-7	3520.0	234	40
94	LA#-7	3729.3	248	20

Fig. 10 – Valori delle note musicali da adottare per il C64.

(segue da fig. 9)

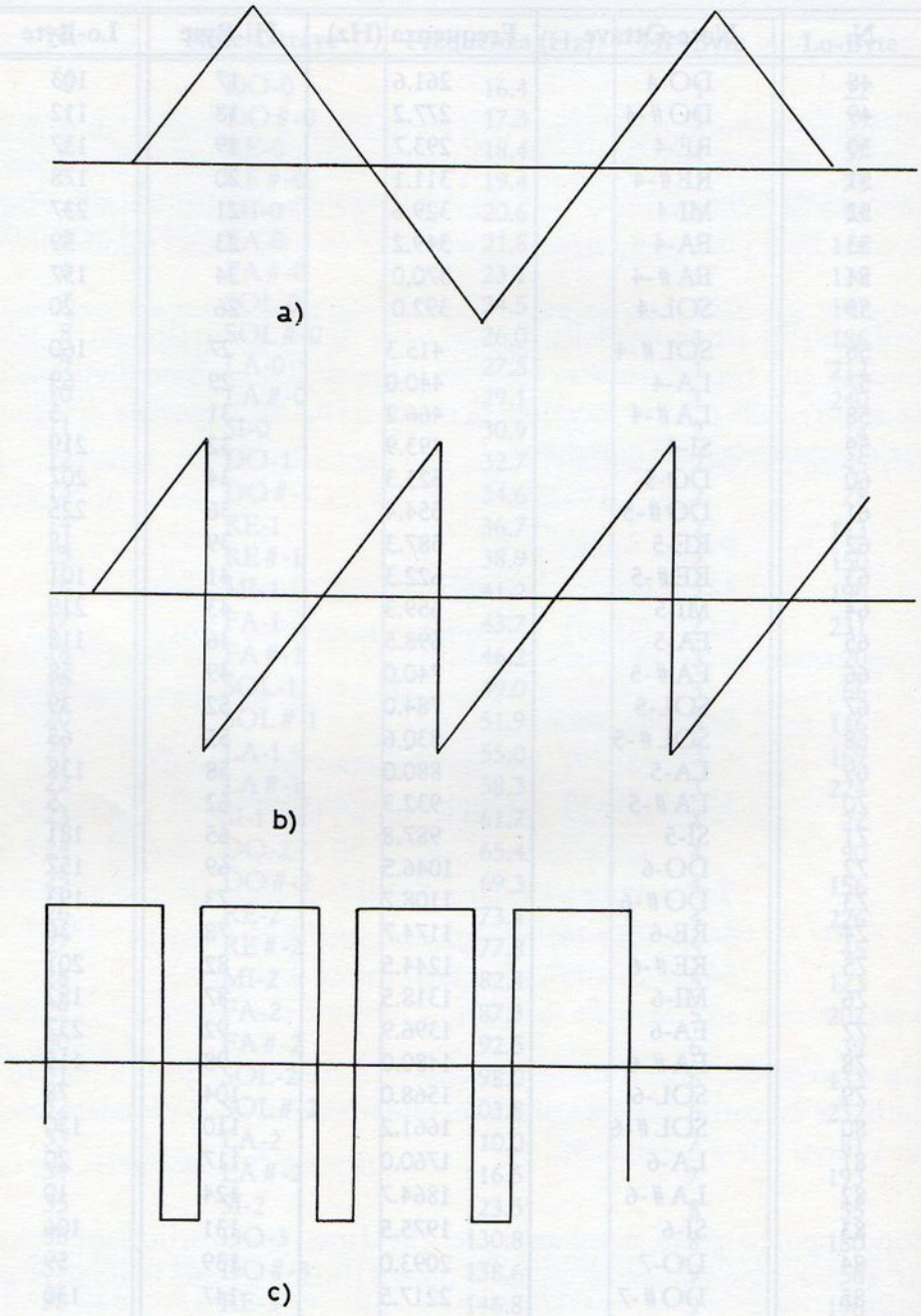


Fig. 11 – Forme d'onda generate dal C64: triangolare (a), a dente di sega (b), e ad onda quadra ad impulso variabile (c).

Essa viene attivata inserendo (**POKE**) nel corrispondente registro del SID il parametro 17.

La seconda forma d'onda è chiamata **a dente di sega** (*fig. 11-b*) e produce un suono prevalentemente metallico. Il relativo parametro da inserire nel

corrispondente registro del SID è 33.

La terza forma d'onda è denominata **onda quadra ad impulso variabile** (*fig. 11-c*) ed il relativo parametro è 65.

Le locazioni di memoria del SID nelle quali vengono immessi, per ciascuna voce, i parametri delle forme d'onda ora descritte sono:

Prima voce
54276 (SI+4) =forma d'onda (17, 33, 65)

Seconda voce
54283 (SI+11)=forma d'onda (17, 33, 65)

Terza voce
54290 (SI+18)=forma d'onda (17, 33, 65)

Adottando la forma d'onda quadra (65) occorre anche definire un parametro addizionale, che indica la durata della parte (impulso) superiore e della parte inferiore dell'onda stessa.

Ciò è ottenuto prendendo in considerazione la parte superiore dell'onda ed utilizzando due locazioni di memoria del SID per ogni voce.

Nella locazione con numero piú basso viene memorizzato un valore compreso fra Ø e 255 (byte basso) e nel registro con numero piú alto un valore compreso fra Ø e 15 (byte alto).

Variando il valore del parametro addizionale dell'onda quadra cosí definito, si possono ottenere suoni simili a quelli dell'organo, oppure suoni analoghi a quelli di un pianoforte o di uno xilofono.

Le locazioni del SID utilizzate per il parametro addizionale dell'onda quadra sono:

Prima voce
54274 (SI+2) =durata onda quadra (byte basso)
54275 (SI+3) =durata onda quadra (byte alto)

Seconda voce
54281 (SI+9) =durata onda quadra (byte basso)
54282 (SI+10)=durata onda quadra (byte alto)

Terza voce

54288 (SI+16)=durata onda quadra (byte basso)
54289 (SI+17)=durata onda quadra (byte alto)

Per quanto riguarda la quarta forma d'onda relativa al rumore, il suo valore è 129 e viene inserito nelle stesse locazioni di memoria utilizzate per le tre forme d'onda prima esaminate.

Suonando una nota musicale con un qualsiasi strumento, si può notare che essa è caratterizzata da quattro diverse fasi, come già precedentemente accennato.

Dapprima il suono sale rapidamente al valore massimo: tale fase viene detta **attacco**.

Successivamente il suono si attenua portandosi al suo valore medio: questa fase prende il nome di **decadimento**.

Il tempo in cui il suono si mantiene attivo al valore medio prende il nome di **sostegno** o **sostentamento**.

Infine, l'ultima fase in cui il suono dal valore medio scende al valore zero viene denominata **rilascio** (*fig. 12*).

Gli stessi effetti sonori ora citati possono essere ottenuti per ciascuna nota prodotta dal C64 grazie al generatore d'inviluppo del suono (ADSR) di cui il C64 è munito.

Le quattro fasi in questione vengono espresse con valori compresi fra Ø e 15 (denominati parametri ADSR), che si inseriscono in due apposite locazioni di memoria del SID per ciascuna delle tre voci del computer, come di seguito indicato:

Prima voce

54277 (SI+5) =attaccare/decadere
54278 (SI+6) =sostenere/rilasciare

Seconda voce

54284 (SI+12)=attaccare/decadere
54285 (SI+13)=sostenere/rilasciare

Terza voce
54291 (SI+19)=attaccare/decadere
54292 (SI+20)=sostenere/rilasciare

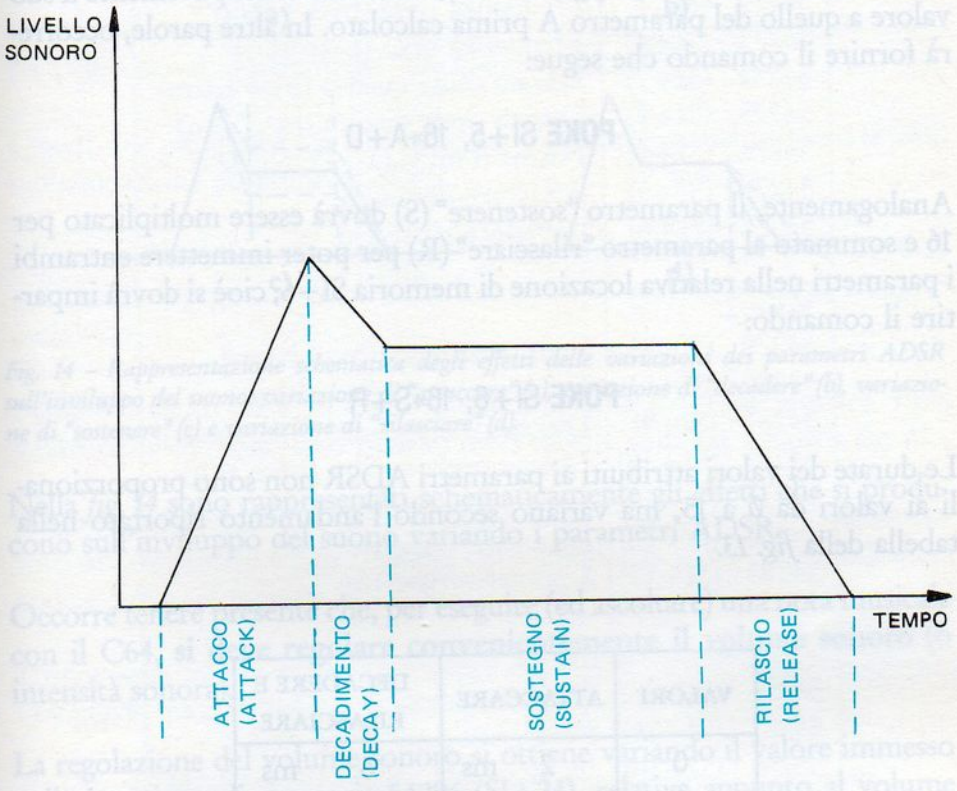


Fig. 12 – Rappresentazione delle quattro fasi che caratterizzano il suono di una nota.

Ciascuna locazione viene suddivisa in due parti costituite da 4 bit e in ognuna di queste parti viene immesso un parametro ADSR.

Vediamo ora come si deve procedere per inserire i valori numerici (variabili da Ø a 15) nelle singole locazioni di memoria prima riportate.

Per meglio chiarire la trattazione, prenderemo in considerazione la prima voce, ossia le locazioni SI+5 e SI+6. Ovviamente, il procedimento è valido anche per le altre due coppie di locazioni di memorie, relative alla seconda ed alla terza voce.

Il parametro "attaccare", che chiameremo A, una volta che ne è stato definito il valore, viene immesso nella parte piú alta del registro SI+5

(primi 4 bit di sinistra). Per fare ciò occorre moltiplicare per 16 (quarta potenza del 2) il valore attibuito ad A.

Il parametro "decadere" (D) viene immesso nella parte bassa dello stesso registro SI+5 (secondo gruppo di 4 bit), sommando semplicemente il suo valore a quello del parametro A prima calcolato. In altre parole, occorrerà fornire il comando che segue:

```
POKE SI+5, 16*A+D
```

Analogamente, il parametro "sostenere" (S) dovrà essere moltiplicato per 16 e sommato al parametro "rilasciare" (R) per poter immettere entrambi i parametri nella relativa locazione di memoria SI+6; cioè si dovrà impartire il comando:

```
POKE SI+6, 16*S+R
```

Le durate dei valori attribuiti ai parametri ADSR non sono proporzionali ai valori da Ø a 15, ma variano secondo l'andamento riportato nella tabella della *fig. 13*.

VALORI	ATTACCARE		DECADERE E RILASCIARE	
0	2	ms	6	ms
1	8	ms	24	ms
2	16	ms	48	ms
3	24	ms	72	ms
4	38	ms	114	ms
5	56	ms	168	ms
6	68	ms	204	ms
7	80	ms	240	ms
8	100	ms	300	ms
9	250	ms	750	ms
10	500	ms	1,5	s
11	800	ms	2,4	s
12	1	s	3	s
13	2	s	9	s
14	5	s	15	s
15	8	s	24	s

Fig. 13 - Durata dei valori "attaccare", "decadere" e "rilasciare".

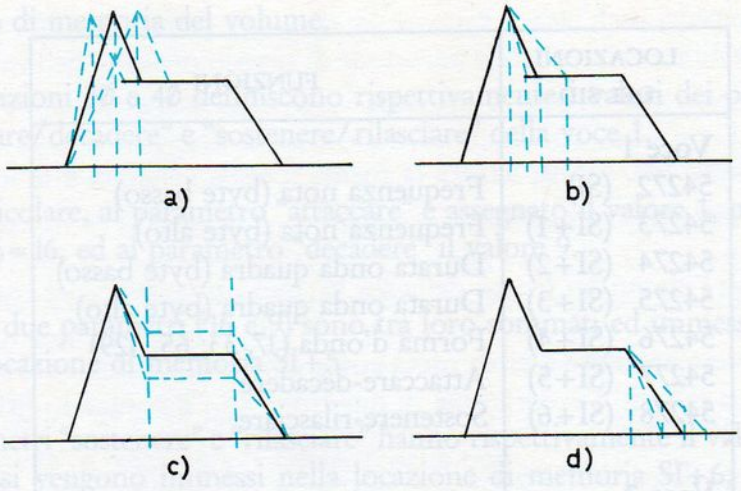


Fig. 14 – Rappresentazione schematica degli effetti delle variazioni dei parametri ADSR sull'inviluppo del suono: variazione di "attaccare" (a), variazione di "decadere" (b), variazione di "sostenere" (c) e variazione di "rilasciare" (d).

Nella *fig. 14* sono rappresentati schematicamente gli effetti che si producono sull'inviluppo del suono variando i parametri ADSR.

Occorre tenere presente che, per eseguire (ed ascoltare) una nota musicale con il C64, **si deve regolare convenientemente il volume sonoro** (o intensità sonora).

La regolazione del volume sonoro si ottiene variando il valore immesso nella locazione di memoria 54296 (SI+24), relativa appunto al volume sonoro. I valori da immettere in tale locazione sono compresi fra Ø (suono non udibile) e 15 (intesità sonora massima).

Nella *fig. 15* è riportata una tabella riassuntiva delle locazioni di memoria del SID e delle relative funzioni che sono state finora esaminate.

A questo punto possiamo accingerci ad elaborare alcuni programmi esemplificativi, che Le consentiranno di verificare praticamente quanto sinora è stato spiegato.

Successivamente vedremo come si potrà utilizzare il computer per suonare un qualsiasi brano musicale.

Il programma riportato nella *fig. 16* produce il suono di una nota musicale (LA) con forma d'onda triangolare e con forma d'onda a dente di sega.

Prima di eseguire questo programma, esaminiamone le fasi operative.

LOCAZIONI DEL SID	FUNZIONI
Voce 1	
54272 (SI)	Frequenza nota (byte basso)
54273 (SI+1)	Frequenza nota (byte alto)
54274 (SI+2)	Durata onda quadra (byte basso)
54275 (SI+3)	Durata onda quadra (byte alto)
54276 (SI+4)	Forma d'onda (17, 33, 65, 129)
54277 (SI+5)	Attaccare-decadere
54278 (SI+6)	Sostenere-rilasciare
Voce 2	
54279 (SI+7)	Frequenza nota (byte basso)
54280 (SI+8)	Frequenza nota (byte alto)
54281 (SI+9)	Durata onda quadra (byte basso)
54282 (SI+10)	Durata onda quadra (byte alto)
54283 (SI+11)	Forma d'onda (17, 33, 65, 129)
54284 (SI+12)	Attaccare-decadere
54285 (SI+13)	Sostenere-rilasciare
Voce 3	
54286 (SI+14)	Frequenza nota (byte basso)
54287 (SI+15)	Frequenza nota (byte alto)
54288 (SI+16)	Durata onda quadra (byte basso)
54289 (SI+17)	Durata onda quadra (byte alto)
54290 (SI+18)	Forma d'onda (17, 33, 65, 129)
54291 (SI+19)	Attaccare-decadere
54292 (SI+20)	Sostenere-rilasciare
54296 (SI+24)	Volume sonoro

Fig. 15 – Schema riassuntivo delle locazioni di memoria del SID e delle loro funzioni esaminate nella presente lezione.

Le istruzioni presenti alla linea di programma 1Ø provvedono a definire l'inizio di memoria del SID (54272), e ad assegnare alla variabile di nome VO la locazione di memoria relativa al volume sonoro (SI+24).

Nell'istruzione 2Ø viene immesso il numero 15 (valore massimo) nel

registro di memoria del volume.

Le istruzioni 3Ø e 4Ø definiscono rispettivamente i valori dei parametri "attaccare/decadere" e "sostenere/rilasciare" della voce 1.

In particolare, al parametro "attaccare" è assegnato il valore 1, per cui si ha 1*16=16, ed al parametro "decadere" il valore 9.

Questi due parametri (16 e 9) sono tra loro sommati ed immessi (**POKE**) nella locazione di memoria SI+5.

I parametri "sostenere" e "rilasciare" hanno rispettivamente il valore 4*16 e 4. Essi vengono immessi nella locazione di memoria SI+6, sempre della voce 1.

Nelle istruzioni 5Ø e 55 viene definita la nota LA-4 da suonare, i cui valori (29 e 69) sono stati ricavati dalla tabella della *fig. 9* e della *fig. 10.*

Il valore della frequenza alta (Hi-Byte) di tale nota (pari a 29) è memorizzato nel registro SI+1 ed il valore della frequenza bassa (Lo-Byte), il cui codice è 69, nel registro SI.

L'istruzione 6Ø provvede ad attivare il generatore dell'onda triangolare (17) per la voce 1 (locazione SI+4).

Il ciclo di ritardo dell'istruzione 7Ø determina la durata della nota.

L'istruzione 8Ø attiva il generatore a dente di sega, sempre con la voce 1,

```
5 REM*SUONO DI UNA NOTA*
10 SI=54272:VO=SI+24
20 POKE VO,15
30 POKE SI+5,16+9
40 POKE SI+6,4*16+4
50 POKE SI,69
55 POKE SI+1,29
60 POKE SI+4,17
70 FOR T=1 TO 500:NEXT
80 POKE SI+4,33
90 POKE SI,69:POKE SI+1,29
95 FOR T=1 TO 500:NEXT
100 POKE SI+4,0:POKE SI+5,0:POKE SI+6,0
```

Fig. 16 – Programma che produce il suono di una nota con forma d'onda triangolare e con forma d'onda a dente di sega.

che provvede anch'essa a suonare la nota LA (istruzione 9Ø), per una durata stabilita dalla seconda iterazione di ritardo presente nell'istruzione 95.

L'istruzione 1ØØ, infine, azzera la forma d'onda generata e le locazioni di memoria relative ai parametri ADSR.

Concluso l'esame del listato della *fig. 16*, può immetterlo nel computer.

Dopo avere regolato a circa metà corsa il volume sonoro del televisore, esegua quindi il programma in questione con il comando **RUN**.

Dall'altoparlante dell'apparecchio televisivo verrà chiaramente riprodotta la nota LA prima con la forma d'onda triangolare poi con la forma

```
10 REM*SCALA MUSICALE*
15 SI=54272
20 VO=54296
25 POKE VO,15
30 POKE SI+5,31
35 POKE SI+6,31
40 FOR N=1 TO 16
45 READ FL,FH
50 POKE SI,FL
55 POKE SI+1,FH
60 POKE SI+4,17
65 FOR T=1 TO 500:NEXT
70 POKE SI+4,0
75 NEXTN
100 DATA 103,17,137,19,237,21,59,23
110 DATA 20,26,69,29,219,32,207,34
120 DATA 207,34,219,32,69,29,20,26
130 DATA 59,23,237,21,137,19,103,17
```

Fig. 17 – Programma che esegue la scala musicale.

d'onda a dente di sega.

Provi ora a modificare il suono riprodotto, variando prima i valori di "attaccare/decadere" (istruzione 3Ø) e poi quelli di "sostenere/rilasciare" (istruzione 4Ø).

Infine, provi a sostituire la nota riprodotta con altre note, immettendo i parametri Lo-Byte e Hi-Byte ad esse relativi nelle corrispondenti locazio-

ni di memoria 5Ø e 55.

Il programma riportato nella *fig. 17* consente invece di riprodurre la scala musicale in DO maggiore ascendente e discendente, sempre con la voce 1.

Le istruzioni 15 e 2Ø assegnano rispettivamente alla variabile SI il valore d'inizio della memoria del SID (54272) ed alla variabile VO il valore della locazione di memoria relativa al volume (54296), il quale viene regolato al massimo con l'istruzione 25.

Le istruzioni 3Ø e 35 definiscono rispettivamente i parametri "attaccare/decadere" e "sostenere/rilasciare" delle note.

In particolare, il valore di "attaccare" è 1, per cui 1*16=16, ed il valore di "decadere" è 15: il valore complessivo della somma dei due parametri è quindi 31.

Anche i parametri "sostenere/rilasciare" hanno un valore complessivo pari a 31 (1*16=16+15=31).

L'istruzione **FOR** della linea di programma 4Ø impone alla **READ** (riga 45) di leggere i valori delle sedici note musicali riportate nelle istruzioni 1ØØ÷13Ø.

Alle variabili FL e FH vengono assegnati rispettivamente i valori Lo-Byte e Hi-Byte di ciascuna nota della scala musicale, valori che sono poi immessi, ad ogni ciclo **FOR**, nelle corrispondenti locazioni di memoria del SID (istruzioni 5Ø e 55).

L'istruzione 6Ø attiva la forma d'onda triangolare per ciascuna nota riprodotta.

L'iterazione di ritardo presente nella linea di programma 65 fornisce il tempo di durata di ciascuna nota.

L'istruzione 7Ø provvede infine a disattivare il generatore della forma d'onda triangolare al termine di ciascuna nota.

Nelle istruzioni **DATA** 1ØØ÷13Ø, come già accennato, sono riportati i valori numerici di seguito indicati, corrispondenti alle note musicali della scala considerata:

NOTA	LO-BYTE	HI-BYTE
DO	1Ø3	17
RE	137	19
MI	237	21
FA	59	23
SOL	2Ø	26
LA	69	29
SI	219	32
DO	2Ø7	34

Dopo aver cancellato lo schermo e la memoria del computer, immetta in quest'ultimo il programma ora descritto e lo esegua con il comando **RUN**.

Dall'altoparlante del televisore verranno emesse le note della scala musicale in DO maggiore ascendente e discendente.

Può provare ad eseguire il programma in questione scegliendo un'altra scala musicale, nel qual caso dovrà sostituire nelle **DATA** 1ØØ÷13Ø i valori numerici attuali con quelli delle note musicali da Lei prescelte.

Nei programmi sinora esaminati sono state utilizzate le forme d'onda triangolare ed a dente di sega per creare le note.

Esamineremo ora un nuovo programma che utilizza la forma d'onda quadra per riprodurre la scala musicale con un suono simile a quello del pianoforte.

Il programma in questione è riportato nella *fig. 18*; come può rilevare, le varie fasi operative sono analoghe a quelle dei precedenti programmi.

In questo caso i valori dei parametri "attaccare/decadere" hanno rispettivamente il valore Ø e 9 (istruzione 4Ø).

Hanno pure valore Ø i parametri "sostenere/rilasciare" (istruzione 5Ø).

La scelta di tali valori è stata determinata dal fatto che si desidera simulare il suono del pianoforte.

Infatti, quando si preme un tasto del pianoforte il suono emesso raggiunge istantaneamente il valore massimo, dal quale discende poi lentamente per tutta la durata in cui il tasto rimane abbassato.

Se si alza invece il dito dal tasto prima che il suono si annulli, la nota

```
10 REM*PIANOFORTE CON C64*
20 SI=54272
25 VO=54296
30 POKE VO,15
40 POKE SI+5,16*0+9
50 POKE SI+6,0
60 POKE SI+3,8
65 FOR N=1 TO 8
70 READ FL,FH
80 POKE SI,FL
90 POKE SI+1,FH
100 POKE SI+4,65
110 FOR T=1 TO 500:NEXT
140 POKE SI+4,0
150 NEXTN
1000 DATA103,17,137,19,237,21,59,23
1010 DATA20,26,69,29,219,32,207,34
```

Fig. 18 – Programma che riproduce la scala musicale con un suono simile a quello del pianoforte.

emessa si interrompe istantaneamente, ossia il volume sonoro cade a zero.

Le istruzioni 55÷6Ø immettono nelle corrispondenti due locazioni di memoria del SID i valori della durata della parte superiore dell'onda quadra.

Piú precisamente, si immettono il valore Ø nella locazione SI+2 (byte basso) ed il valore 8 nella locazione SI+3 (byte alto).

In questo modo la durata della parte positiva dell'onda quadra risulta eguale alla durata della parte negativa.

L'istruzione 1ØØ provvede ad attivare il generatore ad onda quadra (codice 65).

I valori numerici relativi alle otto note musicali della scala che si desidera suonare sono riportati nelle istruzioni **DATA** 1ØØØ÷1Ø1Ø.

Dopo avere cancellato dalla memoria del computer il precedente programma, vi immetta il listato riportato nella *fig. 18*.

Esegua poi il programma con il comando **RUN**: dall'altoparlante del tele-

```
10 REM*SUONIAMO UN ACCORDO*
20 VO=54296
30 S1=54272
40 POKEVO,15
50 POKES1,158
60 POKES1+1,11
70 POKES1+6,16*15
80 POKES1+4,17
90 FORT=1TO2000:NEXT
100 POKES1,162
110 POKES1+1,14
120 FORT=1TO2000:NEXT
130 POKES1,103
140 POKES1+1,17
150 FORT=1TO2000:NEXT
160 POKES1+4,0
170 PRINT:PRINT"PER SUONARE L'ACCORDO PREMI RETURN"
180 INPUTA$
190 POKES1+4,17
200 S2=54279
210 POKES2,158
220 POKES2+1,11
230 POKES2+6,16*15
240 POKES2+4,17
250 S3=54286
260 POKES3,162
270 POKES3+1,14
280 POKES3+6,16*15
290 POKES3+4,17
295 FORT=1TO3500:NEXT
300 POKES1+4,0
310 POKES2+4,0
320 POKES3+4,0
```

Fig. 19 – Programma che suona un accordo musicale con tre note.

visore verranno emesse le note della scala musicale con un suono simile a quello di un pianoforte.

Il programma codificato nella *fig. 19* consente invece di suonare dapprima tre note separatamente e poi di suonarle tutte assieme, in modo da formare un accordo musicale.

L'istruzione 2Ø assegna alla variabile VO la locazione di memoria del volume (54296) e l'istruzione 3Ø attribuisce alla variabile S1 (S uno) il valore d'inizio delle locazioni di memoria del SID (54272).

Le istruzioni 5Ø÷8Ø provvedono a suonare la nota FA-3 (codici 158 e 11) con forma d'onda triangolare (istruzione 8Ø). Tale forma d'onda viene usata anche per tutte le altre note previste dal programma.

Il valore di sostentamento delle note generate è fornito dall'istruzione 7Ø.

L'istruzione 9Ø stabilisce il tempo di durata della nota.

Le istruzioni 1ØØ÷12Ø provvedono a fare suonare la nota LA-3 (codici 162 e 14), mentre le istruzioni 13Ø÷14Ø generano la nota musicale DO-4 (codici 103 e 17).

L'istruzione 16Ø disattiva il generatore della forma d'onda triangolare e l'istruzione 17Ø stampa sullo schermo il messaggio che avverte l'operatore di battere il tasto **RETURN** per ottenere l'accordo musicale di tre note.

Battendo il tasto **RETURN** viene nuovamente attivato il generatore della forma d'onda triangolare (istruzione 19Ø) e quindi viene un'altra volta riprodotta la nota DO-4 (istruzione 13Ø÷14Ø).

Nell'istruzione 2ØØ viene assegnata alla variabile S2 la locazione di memoria 54279, relativa alla seconda voce del SID.

Le istruzioni 21Ø÷24Ø fanno suonare la nota FA-3 (codici 158 e 11), il cui valore di sostentamento è definito dall'istruzione 23Ø.

L'istruzione 25Ø assegna alla variabile S3 la locazione di memoria 54286, relativa alla terza voce del SID.

Le istruzioni 26Ø÷29Ø provvedono a fare suonare la nota musicale LA-3 (codici 162 e 14) con il valore di sostentamento definito dall'istruzione 28Ø.

Infine, le istruzioni 3ØØ÷32Ø disattivano i generatori della forma d'onda triangolare delle tre voci del SID.

Dopo avere cancellato dalla memoria il precedente programma, immetta nel computer il listato della *fig. 19* e lo esegua con il comando **RUN**.

Verranno emesse una dopo l'altra le note DO, FA e LA con un suono simile a quello di un organo, quindi il programma si arresterà e sullo

schermo apparirà il messaggio:

PER SUONARE L'ACCORDO PREMI RETURN

Battendo tale tasto udrà il suono prodotto dall'accordo delle tre note DO, FA, LA.

Nella prossima lezione vedremo come si può suonare un brano musicale con il C64.

5 — Sprite con il C64

Nella precedente lezione ha avuto modo di apprendere la tecnica che viene adottata per creare gli sprite, ossia le immagini ad alta risoluzione grafica.

Approfondiremo ora le conoscenze sul loro impiego mediante la realizzazione di appositi programmi dimostrativi, che possono essere considerati esempi-guida da utilizzare per creare i propri effetti di animazione.

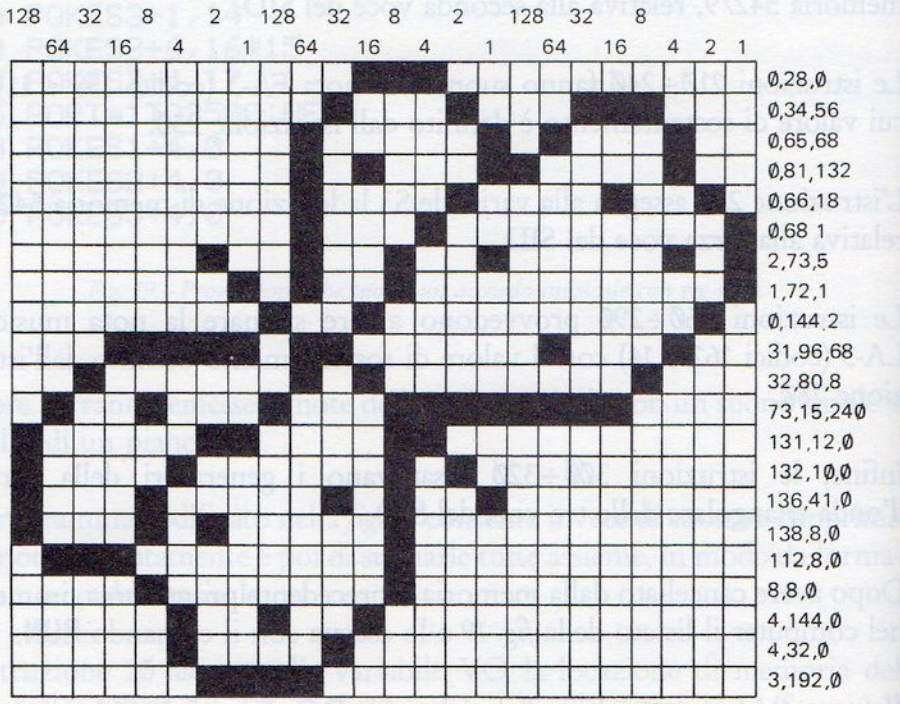


Fig. 20 – Sprite raffigurante una farfalla.

Il programma esemplificativo che ci proponiamo di eseguire consente di visualizzare sullo schermo uno sprite raffigurante una farfalla (rappresentata nella *fig. 20*), che si sposta sullo schermo verso il basso e verso destra.

Il listato del programma in questione è riportato nella *fig. 21*.

```
10 REM**********************
15 REM*PROGRAMMA CON SPRITE*
20 REM**********************
25 PRINT"□"
30 V=53248
40 FOR N=0 TO 62
50 READ Q
60 POKE 832+N,Q
70 NEXT N
90 POKE 2040,13
95 POKE V+39,7
100 POKE V+21,1
120 FOR X=0 TO 255
130 POKE V+0,X
140 POKE V+1,X
150 NEXT X
160 GOTO 120
500 DATA 0,28,0,0,34,56,0,65,68
510 DATA 0,81,132,0,66,18,0,68,1
520 DATA 2,73,5,1,72,1,0,144,2
530 DATA 31,96,68,32,80,8,73,15,240
540 DATA 131,12,0,132,10,0,136,41,0
550 DATA 138,8,0,112,8,0,8,8,0
560 DATA 4,144,0,4,32,0,3,192,0
```

Fig. - 21 - Programma che visualizza sullo schermo del C64 uno sprite raffigurante una farfalla.

Le istruzioni 25÷9Ø svolgono le stesse funzioni descritte nella precedente lezione e che ora riassumeremo brevemente.

L'istruzione 4Ø legge i 63 dati relativi allo sprite, riportati nelle istruzioni **DATA** (5ØØ÷56Ø), i quali vengono memorizzati nell'area di memoria 832 (blocco 13).

L'istruzione 9Ø provvede a posizionare il puntatore dello sprite Ø nella posizione inizio dati (13) dell'area di memoria 832.

L'istruzione 95, infine, colora lo sprite, il quale viene "acceso" mediante l'istruzione 1ØØ.

Le nuove istruzioni 12Ø÷15Ø determinano lo spostamento dello sprite dall'alto in basso e da sinistra a destra dello schermo. Ciò si ottiene mediante l'iterazione presente nella linea di programma 12Ø, che provvede a cambiare i valori della variabile X da Ø a 255.

Tali valori vengono immessi nella locazione delle coordinate dello sprite Ø per determinarne lo spostamento verticale (istruzione 13Ø) ed orizzontale (14Ø).

Cancellato il precedente programma dalla memoria del computer, immetta in quest'ultimo il listato riportato nella *fig. 21* e lo esegua con il comando **RUN**.

Sullo schermo apparirà una farfalla stilizzata, che si sposta in modo continuo dall'angolo superiore sinistro dello schermo verso destra e verso il basso.

Per fermare il movimento dello sprite basta battere il tasto **RUN | STOP**.

Se desidera raddoppiare le dimensioni dello sprite, deve inserire nel registro 23 (espansione verticale dello sprite) e nel registro 29 (espansione orizzontale dello sprite) il valore 1, scrivendo l'istruzione che segue:

```
97 POKE V+23,1: POKE V+29,1
```

Impartendo nuovamente il comando **RUN**, sullo schermo apparirà la farfalla con dimensioni raddoppiate.

Per ottenere lo spostamento orizzontale della farfalla da sinistra a destra dello schermo, occorre attribuire un valore costante alla coordinata verticale dello sprite Ø, modificando l'istruzione 14Ø nel seguente modo:

```
140 POKE V+1,130
```

Eseguendo il programma, vedrà la farfalla spostarsi con continuità da sinistra a destra sullo schermo.

Come certamente potrà notare, lo sprite, spostandosi orizzontalmente, non giunge a toccare il lato destro dello schermo; ciò perché lo schermo ha una larghezza di 320 pixel (punti) ed il registro della direzione orizzontale dello sprite può contenere soltanto un valore massimo pari a 255. Nella lezione successiva vedremo come superare tale ostacolo.

Se infine si vuole fare in modo che lo sprite, anziché spostarsi con continuità, si posizioni a caso, sempre orizzontalmente, sullo schermo, ossia "salti" avanti ed indietro, si **deve eliminare** l'istruzione 15Ø e modificare l'istruzione 12Ø come segue:

```
12Ø X=RND (1)*255:FOR T=1 TO 1ØØØ:NEXT
```

Vediamo ora come si possono fare comparire contemporaneamente altri sette sprite sullo schermo (ricordi che si possono visualizzare al massimo otto sprite per volta).

Per fare ciò, occorre inserire le necessarie istruzioni nel programma precedentemente eseguito.

Le nuove istruzioni dovranno svolgere le seguenti funzioni operative:

- definire i puntatori di ciascuno sprite;
- inserire le coordinate per stabilire le posizioni in cui gli sprite dovranno apparire sullo schermo;
- assegnare il colore ad ognuno di essi;
- attivarli mediante il registro 21.

Il programma che consente di visualizzare contemporaneamente otto sprite è riportato nella *fig. 22*; di esso esamineremo soltanto le nuove istruzioni.

L'istruzione 9Ø prepara gli otto puntatori; ciò si ottiene mediante l'iterazione, presente in tale riga di programma, la quale provvede ad assegnare alla variabile P i valori da Ø a 7, che verranno poi immessi nella locazione di memoria 2040.

Nell'istruzione 1ØØ viene chiamata una subroutine, che inizia dalla riga 2ØØ e termina all'istruzione 27Ø e nella quale sono riportate le coordinate orizzontali e verticali di ciascuno sprite. Ad esempio, nella linea di programma 2ØØ sono assegnate rispettivamente le coordinate Ø (posizione orizzontale) e 1ØØ (posizione verticale) allo sprite Ø; nella linea di programma 21Ø sono assegnate le coordinate 6Ø (posizione orizzontale) e 1ØØ (posizione verticale) allo sprite 1 e cosí via di seguito.

La linea di programma 11Ø abilita tutti gli sprite (da Ø a 7).

```
10 REM*****************************
15 REM*PROGRAMMA CON OTTO SPRITE*
20 REM*****************************
25 PRINT"J"
30 V=53248
40 FOR N=0 TO 62
50 READ Q
60 POKE 832+N,Q
70 NEXT N
90 FOR P=0 TO 7:POKE 2040+P,13:NEXT
100 GOSUB 200
110 POKEV+21,255
120 FOR X=39 TO 46
130 POKEV+X,7
140 NEXT X
150 GOTO 120
200 POKEV+0,30:POKEV+1,100
210 POKEV+2,60:POKEV+3,100
220 POKEV+4,90:POKEV+5,100
230 POKEV+6,120:POKEV+7,100
240 POKEV+8,30:POKEV+9,140
250 POKEV+10,60:POKEV+11,140
260 POKEV+12,90 :POKEV+13,140
270 POKEV+14,120:POKEV+15,140:RETURN
500 DATA 0,28,0,0,34,56,0,65,68
510 DATA 0,81,132,0,66,18,0,68,1
520 DATA 2,73,5,1,72,1,0,144,2
530 DATA 31,96,68,32,80,8,73,15,240
540 DATA 131,12,0,132,10,0,136,41,0
550 DATA 138,8,0,112,8,0,8,8,0
560 DATA 4,144,0,4,32,0,3,192,0
```

Fig. 22 – Programma che visualizza sullo schermo del C64 otto sprite raffiguranti una farfalla.

Il ciclo di programma costituito dalle istruzioni 120÷140 consente di colorare in giallo gli sprite, immettendo il numero 7 nei registri da 39 a 46, riservati appunto a tale funzione.

Come al solito, prima di immettere il nuovo programma nel computer ed eseguirlo, dovrà cancellare il programma precedente.

Sul lato sinistro dello schermo dovranno apparire gli otto sprite, raffiguranti la farfalla, suddivisi in due gruppi di quattro.

Per fare in modo che il colore di ciascuno sprite cambi continuamente,

dovrà modificare le linee di programma 12Ø e 13Ø come segue:

```
12Ø FOR X=Ø TO 7
13Ø POKE V+39+X, RND (Ø)*16
```

Nella prossima lezione avrà modo di apprendere, fra l'altro, come si possono animare gli sprite.

6 — Suoni ed effetti sonori con il VIC-20

Con il VIC-20 è possibile programmare musica, intendendo con il termine "programmare" il fatto che le note musicali possono essere attivate attraverso un programma BASIC.

Il VIC-20 è in grado di creare tre toni musicali (o voci) diversi e generare rumori per effetti speciali, note e rumori che vengono emessi dall'altoparlante dell'apparecchio televisivo collegato al computer stesso.

Il controllo dei suoni e del loro volume (intensità sonora) è ottenuto mediante cinque diverse locazioni di memoria del computer. Modificando il contenuto di tali locazioni, si può attivare la tonalità sonora desiderata e variarne l'intensità.

Le cinque locazioni di memoria riservate al suono del VIC-20, alle quali si accede mediante l'istruzione **POKE**, sono riportate di seguito.

LOCAZIONI	EFFETTO
36874	Tono 1 (tonalità bassa)
36875	Tono 2 (tonalità media)
36876	Tono 3 (tonalità alta)
36877	Rumore
36878	Volume sonoro

Inserendo (**POKE**) in ognuna delle tre locazioni di memoria riservate alle tonalità sonore (36874, 36875 e 36876) il valore numerico compreso fra 128 e 255, si ottiene la riproduzione di tutte le note della scala musicale.

Invece, se si inserisce nella locazione di memoria 36877 un valore compreso fra 128 e 255, si ottengono effetti sonori particolari.

Occorre tenere presente che, per eseguire (ed ascoltare) una nota musicale, il valore immesso nella locazione di memoria 36878, corrispondente al volume sonoro, deve essere sufficientemente alto.

Il valore del volume è dato da un numero compreso fra Ø (suono non udibile) e 15 (intensità sonora massima).

Nella tabella riportata nella *fig. 23* sono elencati tutti i codici numerici approssimativi, corrispondenti alle note musicali da inserire nelle locazioni di memoria riservate alle tonalità sonore.

NOTA	CODICE	NOTA	CODICE
DO	135	SOL	215
DO #	143	SOL #	217
RE	147	LA	219
RE #	151	LA #	221
MI	159	SI	223
FA	163	DO	225
FA #	167	DO #	227
SOL	175	RE	228
SOL #	179	RE #	229
LA	183	MI	231
LA #	187	FA	232
SI	191	FA #	233
DO	195	SOL	235
DO #	199	SOL #	236
RE	201	LA	237
RE #	203	LA #	238
MI	207	SI	239
FA	209	DO	240
FA #	212	DO #	241

Fig. 23 – Tabella riportante i codici numerici corrispondenti alle note musicali del VIC-20.

Come può rilevare, l'insieme delle note è ripartito nelle quattro ottave elencate di seguito.

Codici da 135 a 191 = prima ottava.
Codici da 195 a 223 = seconda ottava.
Codici da 225 a 239 = terza ottava.
Codici da 240 a 255 = quarta ottava.

L'ultima ottava comprende soltanto due note: il DO ed il DO # (DO diesis).

Per riprodurre una nota con il VIC-20, deve in primo luogo impostare il volume sonoro scrivendo ad esempio:

```
POKE 36878, 15
```

locazione volume ——↑ ↑—— valore volume del suono

Successivamente deve impostare un'altra istruzione **POKE**, seguita dall'indirizzo della tonalità musicale e dal codice della nota che desidera ascoltare.

Per esempio, per ascoltare la nota DO # (DO diesis) della terza ottava (corrispondente al codice 227) con tonalità media, dovrà scrivere:

```
POKE 36875, 227
```

locazione tonalità media ——↑ ↑—— codice nota DO #

Dopo avere regolato opportunamente il comando del volume sonoro dell'apparecchio televisivo collegato al computer, udrà dall'altoparlante dello stesso televisore un DO # con tonalità media.

Per interrompere il suono provocato dalla nota musicale di codice 227, occorre "zittire" la tonalità che è stata attivata, e che corrisponde all'indirizzo 36875, scrivendo:

```
POKE 36875,Ø
```

A questo punto è doveroso precisare che, quando si attiva una nota musicale, questa continua a prodursi sino a quando non viene interrotta assegnando il valore Ø all'indirizzo di memoria relativo alla tonalità con la quale viene emessa la nota stessa.

Pertanto, per "spegnere" una nota non è sufficiente annullare il volume sonoro (assegnando il valore Ø alla locazione di memoria 36878), in quanto la nota continuerebbe ad essere emessa anche se non sarebbe piú udibile.

Dopo avere cancellato lo schermo (premendo contemporaneamente i tasti **RUN | STOP** e **RESTORE**) e la memoria del computer (con il comando **NEW**), può ora provare a scrivere e ad eseguire il semplice programma che segue, il quale consente di suonare la nota musicale SOL-3 con ognuna

delle tre diverse tonalità create dal VIC-20: potrà cosí rendersi immediatamente conto delle loro differenti caratteristiche.

```
10 REM*PROVA TONALITA'*
15 REM*  DEL VIC 20   *
20 POKE 36878,15
25 POKE 36874,235
30 FOR T=1 TO 500:NEXTT
35 POKE 36874,0
40 POKE 36875,235
45 FOR T=1 TO 500:NEXTT
50 POKE 36875,0
55 POKE 36876,235
60 FOR T=1 TO 500:NEXTT
65 POKE 36876,0
```

L'istruzione 2Ø di tale programma regola il volume sonoro al valore massimo (15) e l'istruzione 25 attiva la tonalità bassa del VIC-20 con la nota SOL (codice 235).

L'iterazione di ritardo presente nella linea di programma 3Ø fornisce il tempo di durata della nota con tonalità bassa.

L'istruzione 35 provvede a spegnere la nota con tonalità bassa.

In modo analogo operano le linee di programma seguenti, per quanto concerne la nota SOL con tonalità media (istruzione 4Ø) e con tonalità alta (istruzione 55).

Eseguendo il programma (con il comando **RUN**) udrà distintamente la nota musicale SOL riprodotta prima con tonalità bassa o grave, poi con tonalità media, ed infine con tonalità alta.

In base all'esempio fornito, può esercitarsi a suonare altre note musicali nella tonalità desiderata; basterà che inserisca nella relativa locazione di memoria il codice numerico della nota prescelta e si attenga alle indicazioni della tabella della *fig. 23*.

Un esercizio del genere potrà esserLe certamente utile per addestrare l'orecchio ad individuare le diverse note musicali.

Conclusa l'esercitazione, ripulisca lo schermo e cancelli la memoria del computer con il solito sistema.

Il breve programma che segue Le consentirà di riprodurre tutte le 128 note musicali che il computer può generare.

```
10 REM*NOTE MUSICALI*
15 REM*CON IL VIC 20*
20 POKE 36878,15
30 FOR X=128 TO 255
35 POKE 36875,X
40 FOR T=1 TO 100:NEXTT
45 NEXT X
50 POKE 36875,0
55 END
```

Il ciclo **FOR... NEXT...** (istruzione 30÷45) provvede ad inserire i codici numerici delle note musicali nella locazione di memoria 36875 (istruzione 35), relativa alla tonalità media.

Immettendo nel computer il programma in questione ed eseguendolo con il comando **RUN**, udrà tutte le 128 note che il VIC-20 può suonare con tonalità media.

Se poi sostituirà nell'istruzione 35 la locazione di memoria 36875 con la locazione 36874 e poi con la locazione 36876, potrà riascoltare tutte le note del VIC-20 prima con tonalità bassa e poi con tonalità alta.

Le tre tonalità del computer possono essere generate contemporaneamente ed ognuna di esse può durare sino a quando non viene zittita.

E' possibile inoltre, combinando opportunamente le note musicali con le tre diverse tonalità di suoni, suonare qualsiasi brano musicale.

Per riprodurre con il VIC-20 un brano musicale, occorre prendere in considerazione le seguenti variabili:

- **volume sonoro;**
- **tonalità del suono;**
- **nota musicale;**
- **durata.**

Vedremo ora come si può scrivere un programma che consenta di riprodurre brani musicali con il VIC-20.

Naturalmente, non pretendiamo di far diventare il nostro elaboratore un

sintetizzatore musicale completo; esso dovrà limitarsi a suonare correttamente una sequenza di note, che in precedenza si sarà provveduto a caricare nella sua memoria.

Al di là del puro e semplice esercizio, le applicazioni di un tale programma sono svariate. Si possono rendere piú brillanti e piacevoli i giochi realizzati sul sistema, sottolineando le varie fasi con motivetti diversi; si può realizzare una sveglia digitale che ci svegli al mattino al suono di una musichetta anziché con il solito trillo sgradevole ed antipatico; si possono generare suoni d'allarme che viceversa devono essere sgradevoli ed antipatici per attrarre l'attenzione e cosí via.

Per programmare un brano musicale con il VIC-20, occorre in primo luogo codificarlo, ossia passare dai simboli grafici dello spartito musicale (*fig. 8*) ai valori numerici corrispondenti (*fig. 23*).

Per quanto riguarda la durata delle note, si adotterà la stessa codifica usata dai musicisti e riassunta nella tabella della *fig. 8*, prendendo però come intervallo elementare (semi-biscroma) un valore numerico compreso generalmente fra 15 e 30, a seconda del particolare tempo musicale del brano o del proprio gusto musicale.

Per esempio, se si assegna il valore numerico 2Ø alla semibiscroma, la biscroma avrà il valore 2*2Ø=4Ø, la semicroma il valore 4*2Ø=8Ø, la croma il valore 8*2Ø=16Ø e cosí via, per cui si avrà infine per la breve il valore numerico 128*2Ø=256Ø.

Vediamo ora di mettere in pratica quanto precisato, codificando lo spartito musicale riportato nella *fig. 24*, relativo al "Valzer delle candele".

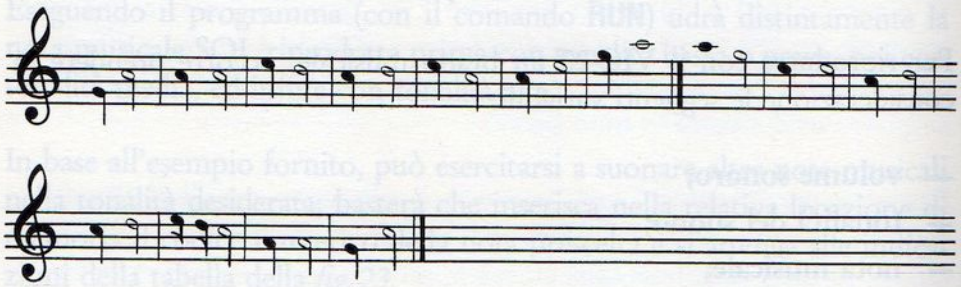


Fig. 24 – Spartito dell'inizio del brano musicale "Valzer delle candele".

Il modo di procedere piú razionale consiste nello scrivere uno di seguito all'altro, in colonna, i codici corrispondenti alle note del brano considerato e nel riportare accanto al codice di ogni nota la relativa durata prescelta. In questo caso, per quanto riguarda la durata delle note, è stato consi-

derato il valore numerico 22,5 per la semibiscroma (intervallo elementare).

Naturalmente il pezzo musicale può essere programmato con un'altra durata delle note, aumentando oppure diminuendo l'intervallo elementare adottato (22,5).

La codifica del motivo relativo al "Valzer delle candele" è riportata di seguito.

CODICE NOTA	DURATA	CODICE NOTA	DURATA
215	36Ø	237	36Ø
225	72Ø	235	72Ø
225	36Ø	231	36Ø
225	72Ø	231	72Ø
231	36Ø	225	36Ø
228	72Ø	228	72Ø
225	36Ø	225	36Ø
228	72Ø	228	72Ø
231	36Ø	231	18Ø
225	72Ø	228	18Ø
225	36Ø	225	72Ø
231	72Ø	219	36Ø
235	36Ø	219	72Ø
237	72Ø	215	36Ø
Ø	72Ø	225	72Ø

Il codice nota Ø con durata 72Ø, presente nelle prime due colonne, indica una pausa d'esecuzione.

I valori numerici codificati vengono poi immessi nella memoria del computer mediante istruzioni-archivio **DATA** nel modo seguente.

```
130 DATA 215,360,225,720,225,360
135 DATA 225,720,231,360,228,720
140 DATA 225,360,228,720,231,360
145 DATA 225,720,225,360,231,720
150 DATA 235,360,237,720,0,720
155 DATA 237,360,235,720,231,360
160 DATA 231,720,225,360,228,720
165 DATA 225,360,228,720,231,180
170 DATA 228,180,225,720,219,360
175 DATA 219,720,215,360,225,720
180 DATA -1
```

Il valore –1 riportato nell'ultima istruzione **DATA** indica la fine dei dati del brano.

E' opportuno scrivere soltanto tre note musicali con relativa durata in ciascuna istruzione **DATA**, al fine di permettere un facile e rapido controllo delle stesse.

Codificato il brano prescelto (nel nostro caso, il "Valzer delle candele"), passiamo ora ad esaminare il programma che ci permette di introdurlo nel computer e di eseguirlo quante volte desideriamo.

Il programma in questione, che consente appunto di riprodurre un brano musicale precedentemente codificato, è riportato nella *fig. 25*.

Tale programma inizia dall'istruzione 4Ø, la quale provvede a regolare al massimo il volume sonoro.

La linea di programma 5Ø impone al computer di leggere (**READ**) le

```
10 REM*PROGRAMMA PER*
20 REM*   SUONARE   *
30 PRINT"□"
35 PRINTTAB(3)"■MUSICA CON VIC 20■"
40 POKE 36878,15
50 READ N
60 IF N=-1 THEN 120
70 READ D
80 POKE 36875,N
90 FOR X=1 TO D:NEXT
100 POKE 36875,0
105 FOR X=1 TO 20 :NEXT
110 GOTO 50
120 END
130 DATA 215,360,225,720,225,360
135 DATA 225,720,231,360,228,720
140 DATA 225,360,228,720,231,360
145 DATA 225,720,225,360,231,720
150 DATA 235,360,237,720,0,720
155 DATA 237,360,235,720,231,360
160 DATA 231,720,225,360,228,720
165 DATA 225,360,228,720,231,180
170 DATA 228,180,225,720,219,360
175 DATA 219,720,215,360,225,720
180 DATA -1
```

Fig. 25 – Programma per eseguire brani musicali con il VIC-20.

informazioni (N) relative alle note da suonare riportate nelle istruzioni **DATA** (istruzioni 13Ø÷18Ø).

All'istruzione 6Ø il programma si chiede se l'ultimo valore letto nelle istruzioni **DATA** è eguale a –1: in caso affermativo l'elaborazione salta all'istruzione 12Ø (fine programma).

Analogamente l'istruzione 7Ø provvede a leggere l'informazione (D) relativa alla durata di ciascuna nota letta in N.

Le note lette con l'istruzione 5Ø vengono inserite (**POKE**) nella locazione di memoria 36875, relativa alla tonalità media del computer (istruzione 8Ø).

In altre parole, l'istruzione 8Ø provvede a suonare la nota N, la quale dura per un tempo D determinato dall'iterazione presente nella linea di programma 9Ø.

L'istruzione 1ØØ spegne la nota, ossia l'annulla per il brevissimo tempo che impiega il computer a "contare" da 1 a 2Ø (istruzione 1Ø5).

L'istruzione 11Ø provvede, infine, a rimandare l'elaborazione alla linea di programma 5Ø.

Il programma ora descritto è valido per suonare qualsiasi brano musicale, modificando ovviamente soltanto le istruzioni DATA.

Dopo avere cancellato lo schermo e la memoria del computer, può immettere in essa il programma listato nella *fig. 25*.

Una volta battuto e dopo aver corretto gli eventuali errori, il programma può essere eseguito con il comando **RUN**.

Dall'altoparlante del televisore verranno emesse le note del "Valzer delle candele".

Provi, ora, a sostituire le istruzioni relative al blocco **DATA** del listato della *fig. 25* con le nuove istruzioni **DATA** riportate nella *fig. 26* e nella *fig. 27*, corrispondenti alla codifica del vecchio motivo popolare "Petit montagnard" (Piccolo montanaro), il cui spartito musicale è mostrato nella *fig. 28*.

Nella prossima lezione illustreremo altre possibilità musicali e sonore del VIC-20.

```
200 DATA 227,240,231,240,227,240
210 DATA 223,240,219,240,219,480
220 DATA 219,240,223,120,227,120
230 DATA 223,240,212,240,215,960
240 DATA 215,240,227,240,231,240
250 DATA 227,240,223,360,228,120
260 DATA 223,360,228,120,223,360
270 DATA 228,120,223,360,227,120
280 DATA 219,960,227,240,231,240
290 DATA 227,240,223,240,219,240
300 DATA 219,480,219,240,223,120
310 DATA 227,120,223,240,212,240
320 DATA 215,960,215,240,227,240
330 DATA 231,240,227,240,223,360
340 DATA 228,120,223,360,228,120
350 DATA 223,360,228,120,227,360
360 DATA 231,120,228,960,0,960
370 DATA 233,240,237,240,233,240
380 DATA 231,240,228,240,233,240
390 DATA 228,240,227,240,223,240
400 DATA 228,240,223,240,227,120
410 DATA 223,120,219,720,227,240
420 DATA 231,240,223,240,227,120
430 DATA 223,120,219,720,227,240
440 DATA 231,240,223,240,227,120
450 DATA 223,120,219,720,233,240
460 DATA 237,240,233,240,231,240
470 DATA 228,240,233,240,228,240
480 DATA 227,240,223,240,228,240
490 DATA 223,240,227,120,223,120
500 DATA 219,720,227,240,231,240
510 DATA 223,240,227,120,223,120
520 DATA 219,720,223,240,227,240
530 DATA 231,240,228,960,0,960
540 DATA 219,240,219,120,223,120
550 DATA 227,120,228,120,231,120
560 DATA 233,120,235,360,239,120
570 DATA 231,360,235,120,227,360
580 DATA 231,120,219,360,227,360
590 DATA 233,360,237,120,228,360
600 DATA 233,120,219,360,228,120
610 DATA 212,360,219,360,215,360
620 DATA 223,120,207,360,215,120
630 DATA 199,360,207,120,183,360
640 DATA 199,360,201,240,212,240
650 DATA 219,240,223,240,219,240
660 DATA 219,120,223,120,227,120
```

Fig. 26 – Blocco dei dati relativi al brano musicale "Petit Montagnard".

```
670 DATA 228,120,231,120,233,120
680 DATA 235,360,239,120,231,260
690 DATA 235,120,227,360,231,120
700 DATA 219,360,227,360,233,360
710 DATA 237,220,228,360,233,120
720 DATA 219,360,228,120,212,360
730 DATA 219,360,215,360,223,120
740 DATA 207,360,215,120,199,360
750 DATA 207,120,183,360,199,360
760 DATA 201,960,-1
```

Fig. 27 – Continuazione del blocco dati relativi al brano musicale "Petit Montagnard".

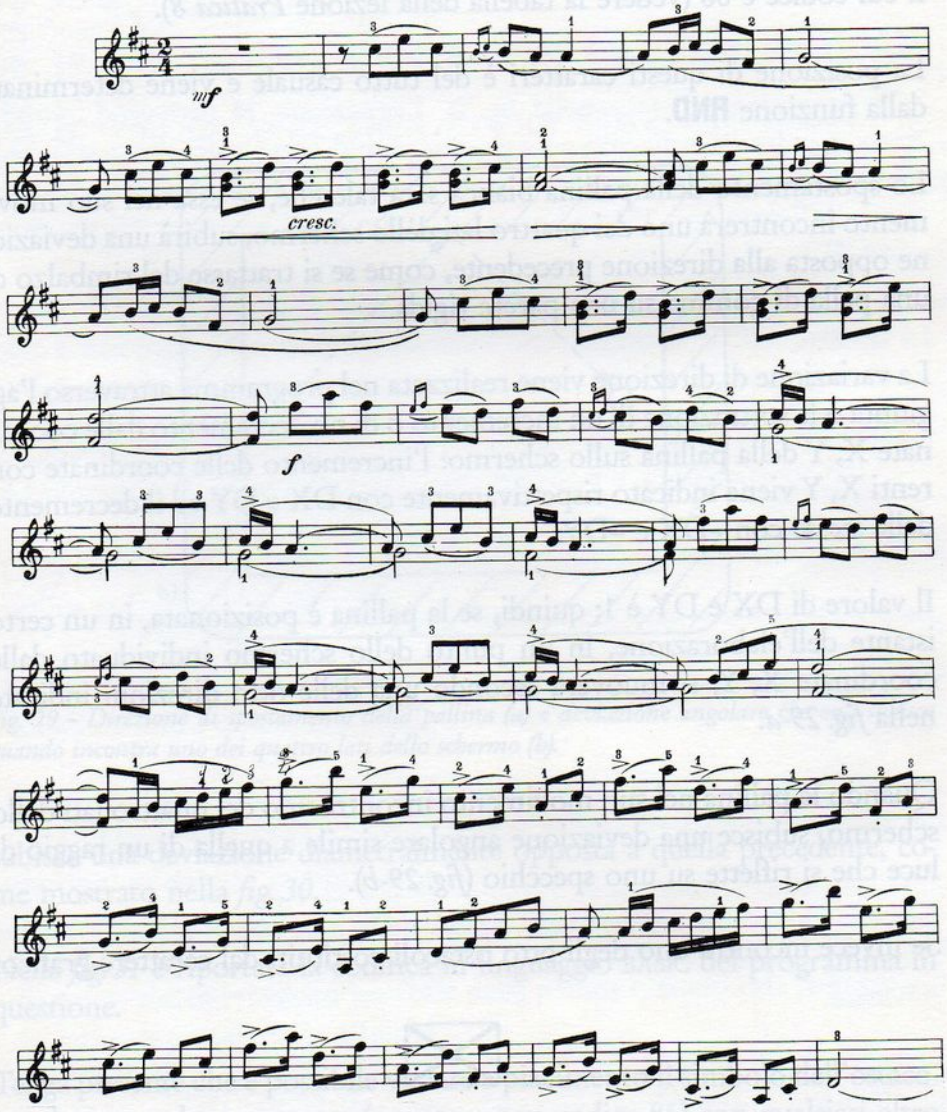


Fig. 28 – Spartito musicale del brano "Petit Montagnard".

7 — Programma d'animazione con il VIC-20

Il semplice programma d'animazione che viene ora proposto provoca il movimento di una pallina bianca (codice carattere 81) in tutte le direzioni su uno schermo nero con il bordo giallo.

Il movimento delle pallina viene contrastato da otto ostacoli, formati dal seguente carattere grafico:

il cui codice è 86 (vedere la tabella della lezione *Pratica 8*).

La posizione di questi caratteri è del tutto casuale e viene determinata dalla funzione **RND**.

Lo spostamento della pallina bianca sarà tale che, se essa nel suo movimento incontrerà uno dei quattro lati dello schermo, subirà una deviazione opposta alla direzione precedente, come se si trattasse del rimbalzo di una palla di gomma su una parete rigida.

La variazione di direzione viene realizzata nel programma attraverso l'aggiunta o la sottrazione di un incremento o di un decremento dalle coordinate X, Y della pallina sullo schermo: l'incremento delle coordinate correnti X, Y viene indicato rispettivamente con DX e DY ed il decremento delle stesse con -DX e -DY.

Il valore di DX e DY è 1; quindi, se la pallina è posizionata, in un certo istante dell'elaborazione, in un punto dello schermo individuato dalle coordinate X, Y, si muoverà secondo una delle otto direzioni indicate nella *fig. 29-a*.

Quando la pallina nel suo movimento incontra uno dei quattro lati dello schermo, subisce una deviazione angolare simile a quella di un raggio di luce che si riflette su uno specchio (*fig. 29-b*).

Se invece incontra uno degli otto ostacoli costituiti dal carattere grafico:

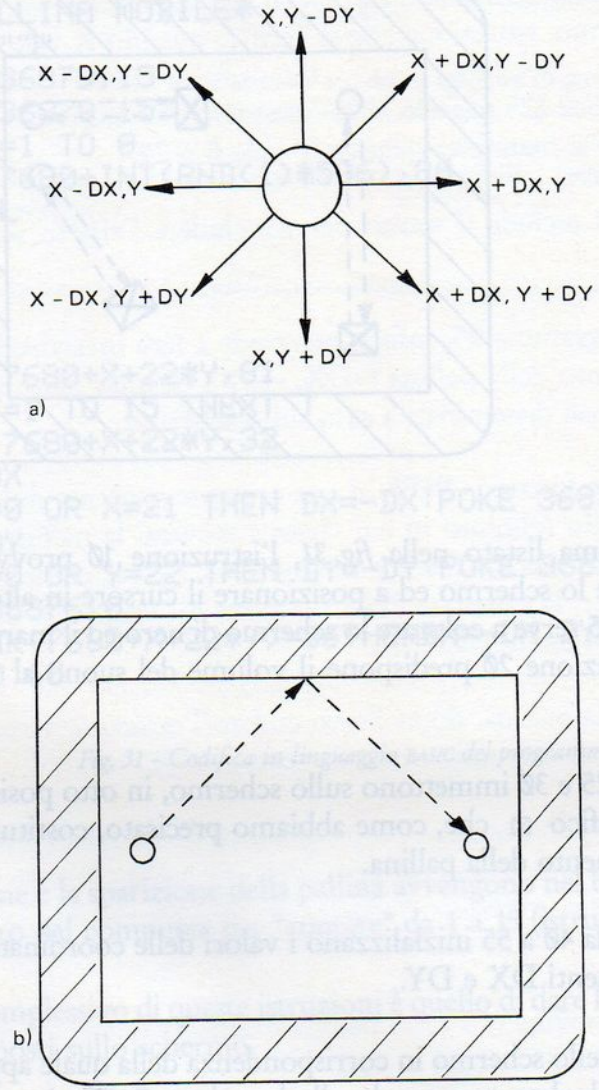


Fig. 29 – Direzione di spostamento della pallina (a) e deviazione angolare che essa subisce quando incontra uno dei quattro lati dello schermo (b).

subisce una deviazione diametralmente opposta a quella precedente, come mostrato nella *fig. 30*.

Nella *fig. 31* è riportata la codifica in linguaggio BASIC del programma in questione.

Tenga presente che è possibile variare a piacimento il simbolo dell'ostacolo adottato nel programma (carattere con codice 86) con qualsiasi altro carattere riportato nella tabella della lezione *Pratica 8*.

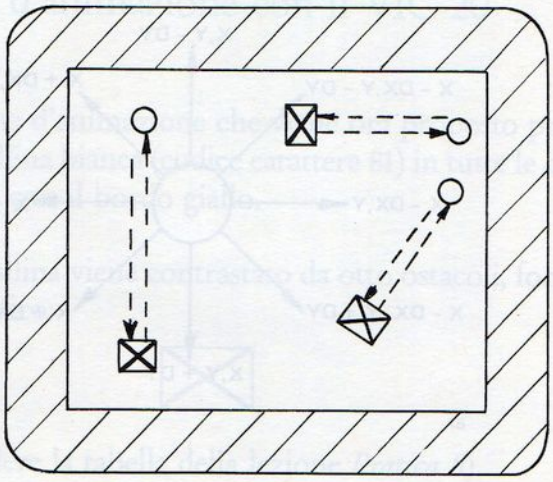


Fig. 30 - Deviazione subita dalla pallina quando incontra uno degli ostacoli.

Nel programma listato nella *fig. 31*, l'istruzione 1Ø provvede, come è noto, a pulire lo schermo ed a posizionare il cursore in alto a sinistra. L'istruzione 15 serve a colorare lo schermo di nero ed il margine di giallo, mentre l'istruzione 2Ø predispone il volume del suono al valore massimo (15).

Le istruzioni 25 e 3Ø immettono sullo schermo, in otto posizioni a caso, il carattere grafico ⊠ che, come abbiamo precisato, costituisce un ostacolo al movimento della pallina.

Le istruzioni da 4Ø a 55 inizializzano i valori delle coordinate X e Y ed i relativi incrementi DX e DY.

La posizione dello schermo in corrispondenza della quale appare la pallina bianca è calcolata sommando alla locazione 768Ø, che coincide con l'angolo superiore sinistro dello schermo, i valori interi ricavati dalla formula:

$$1)\ X+22*Y$$

Ora, per tutti i valori di X variabili da Ø a 21 e per tutti i valori di Y variabili da Ø a 22 avremo tanti valori della precedente formula 1) che, sommati a 768Ø, forniranno appunto gli indirizzi di tutte le caselle dello schermo.

Le istruzioni 6Ø÷7Ø servono a piazzare la pallina (codice 81) nella casella dello schermo corrispondente all'indirizzo calcolato con la formula 1) e subito dopo a cancellarla dallo schermo (codice 32).

```
5 REM*PALLINA MOBILE*
10 PRINT"J"
15 POKE 36879,15
20 POKE 36878,15
25 FOR L=1 TO 8
30 POKE 7680+INT(RND(1)*506),86
35 NEXT L
40 X=1
45 Y=1
50 DX=1
55 DY=1
60 POKE 7680+X+22*Y,81
65 FOR T=1 TO 15 :NEXT T
70 POKE 7680+X+22*Y,32
75 X=X+DX
80 IF X=0 OR X=21 THEN DX=-DX:POKE 36876,219
85 Y=Y+DY
90 IF Y=0 OR Y=22 THEN DY=-DY:POKE 36876,225
93 POKE36876,0
95 IFPEEK(7680+X+22*Y)=86THENDX=-DX:DY=-DY:POKE36876,179:GOTO75
100 GOTO 60
```

Fig. 31 - Codifica in linguaggio BASIC del programma di animazione sul VIC-20.

L'apparizione e la sparizione della pallina avvengono nel brevissimo tempo impiegato dal computer per "contare" da 1 a 15 (istruzione 65).

L'effetto complessivo di queste istruzioni è quello di dare l'illusione che la pallina si sposti sullo schermo.

Le istruzioni 75÷9Ø provvedono a fare rimbalzare la pallina quando questa incontra uno dei lati dello schermo.

In particolare, l'istruzione 8Ø predispone il decremento -DX per invertire la direzione del movimento laterale della pallina quando questa raggiunge uno dei lati verticali dello schermo.

Invece, l'istruzione 9Ø predispone il decremento -DY per invertire la direzione del movimento verticale della pallina quando essa colpisce uno dei lati orizzontali dello schermo.

Il rimbalzo della pallina viene segnalato con l'emissione di una delle due note musicali seguenti:

1) nota di codice 219 quando la pallina tocca entrambi i lati verticali dello schermo, situazione che si verifica quando X vale Ø o 21;

2) nota di codice 225 quando la pallina tocca entrambi i lati orizzontali, evento che si manifesta quando Y vale Ø o 22.

L'istruzione 93 esclude il suono per concludere l'effetto sonoro di un rimbalzo.

La linea di programma 95, infine, provvede a fare invertire la direzione dello spostamento della pallina (-DX, -DY), quando essa incontra uno degli otto ostacoli disseminati a caso sullo schermo.

In particolare, la funzione **PEEK** in essa presente consente di "scoprire" se la posizione dello schermo in cui viene a trovarsi la pallina è occupata oppure no da un ostacolo (codice carattere 86). Infatti, come già precisato precedentemente, la funzione **PEEK** permette di "leggere" il contenuto di una locazione di memoria RAM.

Il rimbalzo della pallina, dopo l'urto contro l'ostacolo, viene segnalato con l'emissione di una nota musicale (codice 179).

A questo punto può immettere il programma in questione nel computer ed eseguirlo con il comando **RUN**: vedrà una pallina bianca rimbalzare contro i lati dello schermo e contro gli ostacoli disposti a caso sullo schermo stesso.

Per fermare il movimento della pallina è sufficiente che prema simultaneamente i tasti **RUN | STOP** e **RESTORE** .

* * * * * * *

In seguito...

Nella prossima lezione verificherà il modo di operare di particolari funzioni che servono per il trattamento delle stringhe.

Eseguirà, inoltre, programmi riguardanti le possibilità grafiche e musicali del computer.

CONTROLLO DIDATTICO 10

Domande

1) - Che cosa sono ed a che cosa servono le subroutine?

.....

.....

.....

.....

.....

2) - Come si utilizzano le subroutine?

.....

.....

.....

.....

.....

.....

3) - I dati memorizzati in una tabella (vettore o matrice) come sono organizzati?

.....

.....

.....

.....

.....

.....

4) - Di quale tipo devono essere i dati memorizzati in una tabella?

.....

.....

.....

.....

5) - Scrivere un programma che calcoli e stampi il valore di Y della funzione $Y=(X/2+3,56)\cdot X^2$. Il valore di X deve essere introdotto mediante la tastiera del computer.

.....

.....

.....

.....

.....

.....

.....

6) - Indicare l'errore presente nella sequenza di programma sotto riportata:

```
10 DEF FN Z (X)=(X↑2+X↑3)*7.2
15 INPUT X
20 Z=FN Y (X)
 :
```

. . . . . .

. . . . . .

. . . . . .

. . . . . .

7) - **Computer C64** – Con quale numero iniziano le locazioni di memoria per la sintesi musicale?

Computer VIC-20 – Quante sono le locazioni di memoria riservate al suono?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

8) - Fra quali valori è compreso il numero utilizzato per il controllo del volume sonoro?

. . . . . .

. . . . . .

9) - Quali sono le istruzioni di programma che consentono la riproduzione delle note musicali?

. . . . . .

. . . . . .

.....

.....

10) - **Computer C64** – Scrivere ed eseguire un programma che suoni le note della scala ascendente e discendente della quinta ottava, utilizzando i seguenti parametri ADSR:A=10; D=3; S=10; R=4 con onda a dente di sega. Si dovrà udire un suono simile a quello di un violoncello.

Computer VIC-20 – Scrivere ed eseguire un programma che suoni le note della scala ascendente e discendente della seconda ottava. La durata delle note deve essere stabilita da un ciclo di ritardo durante il quale il computer "conta" da 1 a 5ØØ.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Risposte

1) - Le subroutine servono ad elaborare parti ripetitive di programma e vengono normalmente richiamate da punti diversi di un programma principale; esse consentono anche una ripartizione del programma stesso in moduli.

2) - Le subroutine vengono richiamate attraverso apposite istruzioni di salto come:

GOSUB...; IF... THEN... GOSUB; ON... GOSUB....

3) - I dati memorizzati in una tabella di memoria (vettore o matrice) sono organizzati sequenzialmente. Nel caso di matrici (tabella a due indici) vengono prima memorizzati i dati della prima riga, poi quelli della seconda, quindi della terza e cosí via di seguito.

4) - I dati memorizzati in una tabella, ad uno o due indici, devono essere di tipo omogeneo; devono essere, cioè, o tutti numeri o tutte stringhe.

5) - Il programma che calcola e stampa sullo schermo il valore di Y della funzione $Y=(X/2+3,56)\cdot X^2$ è il seguente:

```
10 DEF FN Y (X)=(X/2+3.56)*X↑2
20 INPUT X
30 Y=FN Y (X)
40 PRINT Y
50 END.
```

6) - Nella sequenza di programma indicata risulta errata l'istruzione 2Ø, in quanto il nome della funzione da essa richiesta (**FN** Y) non corrisponde al nome della funzione definita nell'istruzione 1Ø (**FN** Z).

7) - **Computer C64** – Le locazioni di memoria per la sintesi sonora iniziano dal numero 54272 ed arrivano sino a 54296.

Computer VIC-20 – Al suono sono riservate le seguenti locazioni di memoria: 36874 (TONO 1); 36875 (TONO 2); 36876 (TONO 3); 36877 (RUMORE); 36878 (VOLUME).

8) - Il valore del volume sonoro è compreso fra Ø (silenzio) e 15 (valore massimo).

9) - Le istruzioni che consentono la riproduzione delle note musicali sono le **POKE**, seguite da un indirizzo di memoria e da un codice numerico corrispondente alla nota che si vuole suonare.

10) - Per il C64 il programma può essere cosí listato:

```
10 SI=54272
20 VO=54296
25 POKE VO,15
30 POKE SI+5,16*10+3
35 POKE SI+6,16*10+4
40 FOR N=1 TO 16
45 READ FL,FH
50 POKE SI,FL
55 POKE SI+1,FH
60 POKE SI+4,33
65 FOR T=1 TO 500:NEXT
70 POKE SI+4,0
75 NEXT N
200 DATA 207,34,18,39,219,43,118,46
210 DATA 39,52,138,58,181,65,157,69
220 DATA 157,69,181,65,148,58,39,52
230 DATA 118,46,219,43,18,39,207,34
```

Per il VIC-20 il programma richiesto può essere cosí listato:

```
10 PRINT"□"
20 POKE 36878,15
30 READ N
40 IF N=-1 THEN 100
50 POKE 36875,N
60 FOR X=1TO500:NEXT
70 POKE 36875,0
90 GOTO 30
100 END
200 DATA 195,201,207,209,215,219,223,225
210 DATA 223,219,215,209,207,201,195,-1
```

* * * * * * * *

PARLA BASIC

Corso teorico-pratico per corrispondenza di

ANALISI E PROGRAMMAZIONE BASIC

Scuola Radio Elettra

10126 Torino - Via Stellone 5

Tel. (011) 674432